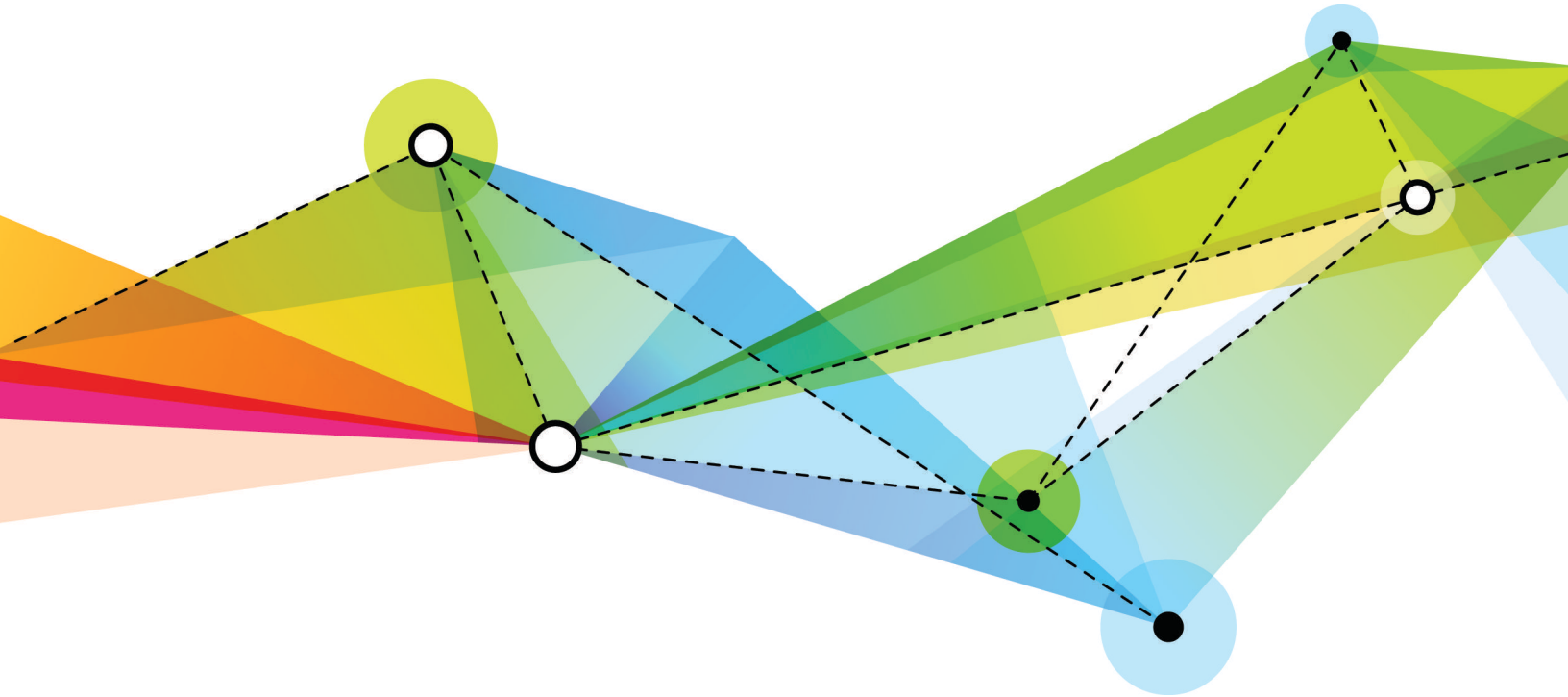
# Smile Developer Guide



inomial

Smile Developer Guide
Edition: 3.10
Release date: July 9, 2018
Smile version: 6.1

Published by Inomial Pty Ltd
Suite 801, 620 Bourke St, Melbourne, Vic 3000, Australia
www.inomial.com • +61 3 9663 3554 • sales@inomial.com • support@inomial.com

Copyright © 2018, Inomial Pty Ltd. Commercial in confidence.

# Contents

**What's changed** ..... **13**

**Chapter 1: Introducing Smile** ..... **14**
- Who this guide is for ..... 14
- Introduction to Smile ..... 14
- Get support ..... 14
- Documentation conventions ..... 15

**Chapter 2: Getting started** ..... **16**
- API features and limitations ..... 16
- Terminology ..... 17
- API components ..... 18
  - API architecture ..... 18
  - The SOAP API ..... 18
  - The message bus ..... 19
  - Example: SOAP methods and the message bus ..... 20
- Set up the Smile API ..... 21
  - Plan your deployment ..... 21
  - Configure SOAP access ..... 21
  - Connect to the SOAP API ..... 22
  - Connect to the message bus ..... 23
- Test the Smile API ..... 24
  - The Java SOAP client ..... 24
  - `echo()` ..... 26
- Write applications ..... 27
  - Best practices ..... 27
  - WSDL and XSD locations ..... 27
  - Namespaces ..... 28
  - SOAP envelopes ..... 29
- Integrate Smile ..... 34
  - Custom fields ..... 34
  - Custom collections ..... 35
  - Accessory data ..... 35
- Recommendations after a Smile upgrade ..... 36

**Chapter 3: Placing orders** ..... **37**
- Order API overview ..... 37
- How to use the order API ..... 38
- Configure a package for API ordering ..... 39
- Retrieve order forms ..... 40

getAvailableOrderForms() ..... 40
getOrderForm() ..... 43
Place an order ..... 46
orderAccount() ..... 46
orderSubscription() ..... 48
reserveUsns() ..... 51
Approve an order ..... 53
approveOrder() ..... 53
Retrieve order information ..... 54
getOrder() ..... 54
Update an order ..... 56
updateOrder() ..... 56
Managing order groups ..... 58
createOrderGroup() ..... 58
addOrderToGroup() ..... 60
placeOrderGroup() ..... 62
getOrderGroup() ..... 65
cancelOrderGroup() ..... 67
cancelOrder() ..... 68
getOrderGroupQuote() ..... 69
Synchronise order information ..... 72
Message sequence—Account orders ..... 72
Message sequence—Subscription orders ..... 73
Message sequence—Provisioning ..... 74

**Chapter 4: Managing accounts ..... 75**
Account API overview ..... 75
Account structure ..... 76
Order an account ..... 77
Retrieve account information ..... 78
get() ..... 78
getAccountInfo() ..... 80
findAccount() ..... 81
findAccounts() ..... 84
getAccountSummary() ..... 87
Manage accounts ..... 89
update() ..... 89
Retrieve transaction information ..... 92
getTransaction() ..... 92
getTransactionDocument() ..... 94
getTransactionList() ..... 95
Download invoice activity statements ..... 98
getRatings() ..... 98
Manage account charges ..... 100
compileChargeList() ..... 100

raiseInvoice() ..... 102
allocate() ..... 108
migrateOpenTransaction() ..... 111
raiseAdjustment() ..... 113
getDeferredCharges() ..... 116
addDeferredCharge() ..... 119
raiseReimbursement() ..... 121
reverseTransaction() ..... 123
Manage cost centres ..... 125
createCostCentre() ..... 125
getCostCentre() ..... 126
getCostCentres() ..... 127
findCostCentre() ..... 128
updateCostCentre() ..... 129
deleteCostCentre() ..... 130
Manage invoice grouping ..... 131
createInvoiceGrouping() ..... 131
getInvoiceGrouping() ..... 133
getAvailableInvoiceGroupings() ..... 134
updateInvoiceGrouping() ..... 136
deleteInvoiceGrouping() ..... 138
Manage purchase orders ..... 139
createPurchaseOrder() ..... 139
getPurchaseOrder() ..... 141
listPurchaseOrders() ..... 142
updatePurchaseOrder() ..... 143
Retrieve future billing estimates ..... 145
getFutureBilling() ..... 145
Manage bonds ..... 149
getBondTypes() ..... 149
getHeldBonds() ..... 150
createBond() ..... 152
releaseBond() ..... 154

**Chapter 5: Managing subscriptions ..... 156**
Subscription API overview ..... 156
Order a subscription ..... 157
Retrieve subscription information ..... 157
findUsername() ..... 157
get() ..... 158
getAuthenticationCredentials() ..... 160
findSubscription() ..... 161
findSubscriptions() ..... 163
getConnectionHistory() ..... 167
getActiveFeatures() ..... 169

Manage subscriptions....171
update()....171
changeUsername()....174
setClearPassword()....175
cancel()....176
suspend()....177
resume()....179
applyFeature()....180
applyFeatureToConnections()....182
stopApplyingFeature()....183
setCostCentre()....184
migrateSubscription()....185
Manage plans....186
getPlanSchedules()....186
getPlanScheduleItemsAt()....187
getPlanChargeDescription()....189
changePlan()....193
Manage real-time rating plan changes....196
getSubscriptionPlans()....196
getServicePlans()....198
proposePlanChange()....200
commitPlanChange()....204
Manage pool reservations....210
getPoolReservationLinks()....210
addIpRangeReservation()....211
removePoolReservationLink()....213
Manage IP ranges....215
getIpRanges()....215
addIpRange()....217
removeIpRange()....219
Download unrated activity statements....220
getActivityStatementRange()....220
Download activity statements in batches....223
getActivityStatementBatch()....224
Download rated activity statements....226
Rating and invoicing cycles....226
getRatingPeriods()....227
getRatingsForPeriod()....229
Synchronise subscription information....231

**Chapter 6: Managing subservices....232**
Subservice API overview....232
Retrieve subservice information....233
getSubserviceBySubserviceUsername()....233
getSubservicesForSubscription()....234

getSubservice() ... 236
Manage subservices ... 237
addSubservice() ... 237
addSubservices() ... 239
removeSubservice() ... 242
updateSubservice() ... 242

**Chapter 7: Managing managed plans ... 244**
Managed plans API overview ... 244
Retrieve managed plan information ... 245
getCurrentManagedPlanGroup() ... 245
getManagedPlanGroup() ... 246
getManagedPlanGroups() ... 248
getManagedPlanChoices() ... 249
getPlanSchedule() ... 254
Manage managed plans ... 256
changePlan() ... 256
changeManagedPlanGroup() ... 260

**Chapter 8: Managing usage and spend ... 262**
Spend management API overview ... 262
Alert thresholds ... 263
Retrieve value pool states ... 263
getSubscriptionValuePoolStates() ... 263
getSubscriptionValuePoolState() ... 265
Retrieve usage alert group states ... 267
getSubscriptionUsageAlertGroups() ... 267
getSubscriptionUsageAlertGroup() ... 269
Retrieve spend limit states ... 271
getSubscriptionSpendLimitStates() ... 271
getSubscriptionSpendLimitState() ... 273
Set spend limit overrides ... 275
setSpendLimitOverride() ... 275
setTemporarySpendLimitOverride() ... 276
setSpendLimitAlertThresholdOverrides() ... 278
Listen for spend limit changes ... 280

**Chapter 9: Managing payments ... 281**
Payments API overview ... 281
Retrieve payment information ... 282
getPayment() ... 282
getPaymentTypes() ... 284
getPendingPayments() ... 285
getAutoPaymentOption() ... 287

Manage payments....289
computeSurcharge()....289
processPayment()....291
validatePayment()....296
recordFailedPayment()....297
reschedulePaymentWithNewDetails()....300
cancelScheduledPayment()....302
Manage stored payment details....304
getStoredPaymentDetails()....304
getStoredPaymentDetailsList()....306
addStoredPaymentDetails()....307
updateStoredPaymentDetails()....309
deleteStoredPaymentDetails()....311

**Chapter 10: Managing prepaid....312**
Prepaid API overview....312
Use prepaid vouchers....313
Voucher use cases....313
validateVoucher()....314
redeemVoucher()....315
Use prepaid blocks....317
getPrepaidState()....317
addPrepaid()....318
updatePrepaid()....320
Use self-serve prepaid....322
getSelfServePrepaidList()....322
getPrepaidPaymentTypes()....323
getQuote()....325
addSelfServePrepaid()....327

**Chapter 11: Managing contracts....329**
Contract API overview....329
Retrieve contract information....330
getContractsForService()....330
getActiveSubscriptionContract()....332
getAvailableContractsForPlan()....333
getCurrentSubscriptionContract()....335
getSubscriptionContractById()....336
getSubscriptionContracts()....337
Manage contracts....340
addSubscriptionContract()....340
updateSubscriptionContract()....342
proposeContractTermination()....344
commitContractTermination()....346

**Chapter 12: Managing custom fields.....349**
Custom fields API overview.....349
Retrieve custom field information.....350
`getCollectionEntry()`.....350
`getCollectionEntryByEntryCode()`.....351
`findCollectionEntries()`.....352
Manage custom fields.....355
`addCollectionEntry()`.....355
`updateCollectionEntry()`.....357
`deleteCollectionEntry()`.....359

**Chapter 13: Managing tickets.....360**
CRM API overview.....360
Retrieve ticket information.....361
`getTicket()`.....361
`getTickets()`.....362
`getTemplateTicketDescription()`.....363
`getAppointmentAvailabilities()`.....365
Manage tickets.....367
`raiseTicket()`.....367
`runTicketAction()`.....370
`addTicketAttachment()`.....371
`updateTicket()`.....372

**Chapter 14: Listening for financial transactions.....374**
Open item accounts.....374
Transaction overview.....374
Transaction types.....376
Reversals.....378
Transaction lifecycle.....379
Synchronise transaction information.....380

**Chapter 15: Performing other Smile tasks.....381**
Endpoint overview.....381
Authenticate credentials.....382
`authenticate()`.....382
Manage accounts receivable.....384
`getItemDescription()`.....384
`getItemDescriptionById()`.....385
Manage CAT.....387
`triggerCatForgotPasswordEmail()`.....387
Manage provisioning.....388
`getIncompleteProvisionRequests()`.....388

provisioningUpdate() 390
Manage recurring charges 392
getSimpleRecurringCharge() 392
getSimpleRecurringCharges() 393
updateSimpleRecurringCharges() 395
addSimpleRecurringCharges() 396
Manage services 399
getServiceDescription() 399
qualifyService() 400
getPlanConfiguration() 402
getTariffTypes() 408
getOverrideableDestinations() 409
Manage number pools 411
search() 411
getLink() 413
createReservation() 414
createLinkToReservation() 416
unlink() 418

**Appendix A: SOAP methods 419**
Account endpoint 419
Accounts receivable endpoint 425
Authentication endpoint 426
CAT endpoint 427
Contract endpoint 428
CRM endpoint 431
Custom fields endpoint 434
Invoice grouping endpoint 436
Managed plans endpoint 437
Order endpoint 440
Payment endpoint 443
Plan change endpoint 447
Pool endpoint 448
Prepaid endpoint 450
Provisioning endpoint 453
Recurring charge endpoint 454
Service endpoint 455
Spend management endpoint 457
Subscription endpoint 460
Subservice endpoint 466

**Appendix B: Message bus topics 468**
SmileAccount 468
AccountUpdated 468
SmileCharge 471

ChargeDeferred....471
SmileConnection....473
ConnectionFeaturesUpdated....473
SmileCustomCollection....475
CustomCollectionEntryAdded....475
CustomCollectionEntryUpdated....476
CustomCollectionEntryDeleted....478
SmileTicket....479
CorrespondenceSent....479
SmileNumberPool....482
LinkDeallocation....482
SmileOrder....484
OrderApproved....484
OrderCancelled....485
OrderComplete....486
OrderEntered....487
OrderRequiresApproval....488
SmileSubscription....489
CreditLimitExceeded....489
CreditLimitOkay....491
PrepaidExpiryWarningEvent....492
RatingConfigurationUpdated....493
SubscriptionActivated....494
SubscriptionClosed....496
SubscriptionEnabled....497
SubscriptionFeaturesChanged....498
SubscriptionIpRangesUpdated....499
SubscriptionPlanChanged....502
SubscriptionProvisionCommand....503
SubscriptionProvisioned....505
SubscriptionProvisionResponse....506
SubscriptionReadyToProvision....507
SubscriptionResumed....508
SubscriptionSpendLimitExceeded....509
SubscriptionSpendLimitReceded....511
SubscriptionSpendLimitStatusUpdated....512
SubscriptionSuspended....514
SubscriptionUpdated....516
SmileTransaction....517
TransactionAllocated....517
TransactionClosed....519
TransactionDeleted....521
TransactionReopened....522
TransactionReversed....524
TransactionUpdated....525

**Appendix C: Faults common to all SOAP methods..... 529**
`AuthenticationException`.....529
`AccessException`.....529

**Appendix D: Smile architecture.....530**

**Appendix E: Support.....531**

**Glossary.....532**

# What's changed

This document was last released as the Smile Developer Guide - Edition 3.9 in May 2018.

The methods and functionality included in the Smile Developer Guide - Edition 3.10 are compatible with **Smile Version 6.0**.

## Updated functionality

Smile Developer Guide - Edition 3.10 updates the following methods and API information:

| Endpoint or topic | Method or topic updated |
|---|---|
| account | • Updated Account API overview |

# Chapter 1

# Introducing Smile

## Who this guide is for

This Developer Guide is for developers who write applications that programmatically interact with Smile, such as applications that:

- integrate Smile into your existing enterprise architecture
- provision customers
- support business-to-business (B2B) and machine-to-machine (M2M) transactions
- create quotes and add prepaid blocks for prepaid and self-serve wireless hotspots

For information about how to use Smile, see the *User Guide*.

## Introduction to Smile

Smile is a business support system (BSS) for ISPs, telcos and utility providers. You can use Smile with many consumer data services, including xDSL and wireless internet, mobile and fixed-line telephony, web hosting, web application management, fax gateways and IPTV.

As a business support and order-to-cash system, Smile:

- creates and tracks invoices on a cycle that suits your company
- sends the invoices to customers
- accepts payments, including credit card and direct debit payments
- automates debtor control
- supports a variety of plans and packages so you can bundle services to customers

Smile integrates with operational support systems (OSS) via well-defined interfaces. Smile can:

- provision and deprovision customers
- automatically control a customer's access to a service based on the status of their account
- enforce spend limits and other policy actions

## Get support

For information about how to contact Inomial, including how to access 24-hour emergency support, see Support.

## Documentation conventions

The following table lists the documentation conventions used in this guide.

**Table 1: Documentation conventions**

| Convention | Meaning |
|---|---|
| **bold** | In text, specifies a user interface element. |
| *italic* | • In code and syntax, specifies a variable value that you supply<br>• In text, specifies a reference or link to another resource |
| underlined | Specifies a default value. |
| `monospaced font` | Specifies system input or output, for example:<br>• method or message names<br>• method signatures<br>• examples |
| [square brackets] | Specifies an optional parameter. |
| *`smile.example.com:port`* | Specifies the URL of your Smile implementation. |
| *`stage`* | Specifies the database of your Smile implementation. For example, *live* or *test*. |

# Chapter 2

# Getting started

## API features and limitations

The API is platform-neutral and supports a subset of Smile functionality. The Smile API consists of endpoints that let you perform the following tasks:

- **account**—get account and transaction information, raise adjustments and reimbursements, manage other account charges, download activity statements, update accounts and manage bonds
- **accounts receivable**—get information about item descriptions
- **authentication**—authenticate credentials passed to Smile by a trusted application
- **CAT**—retrieve and email CAT user password details
- **contracts**—get service and subscription contract information, and add, update and terminate subscription contracts
- **credit card payments**—validate credit card details
- **CRM**—create and retrieve tickets, update tickets and get appointment availability
- **invoice grouping**—get, create and delete invoice groups
- **managed plans**—get managed plan information and change managed plans and groups
- **orders**—reserve usernames, get order forms, order and approve accounts and subscriptions and manage order groups
- **number pools**—get, create and remove number pool reservations
- **plan change**—get service and subscription plans, propose and commit plan changes
- **prepaid subscriptions**—validate and redeem vouchers, provide quotes and add prepaid blocks to subscriptions
- **provisioning**—update the status of provisioning requests
- **recurring charge**—get information about recurring charges and add and update recurring charges
- **services**—qualify services and get service information
- **spend management**—get information about the state of value pools, usage alert groups and spend limits
- **subscriptions**—update subscriptions, change a subscription's plan and get subscription information, including data from custom fields, and manage pool and IP range reservations
- **subservice**—get subservice information, and add, update and remove subservices

In addition, Smile sends messages to the message bus that let you respond to changes in Smile and synchronise Smile data across your network.

However, the API does not support all the functionality you can access in Smile, including:

- entering payments (except adjustments)
- creating packages and plans
- managing workflows or helpdesk tickets
- creating reports

## Terminology

Many BSS applications use the same term in slightly different ways. To avoid confusion, the following list explains what some common terms mean in Smile.

| | |
|---|---|
| **service** | A generic kind of product that can be provided to customers, for example, ADSL or mobile telephony. |
| **subscription** | A single instance of a service that is provided to a customer, for example, a single mobile number. Subscriptions belong to an account and are often the source of usage and recurring charges. |
| **account** | A customer, company or debtor in the accounts receivable system to whom one or more subscriptions are provided. For example, you can provide mobile subscription 0401234567 to the account of Joe Smith. The account defines who bills are sent to, how much the customer owes and their payment terms. |
| **plan** | The set of rules that determine how much is charged for a subscription. A plan can include recurring charges, usage charges, minimum charges, included usage and access rules such as modem speed. |
| **package** | A bundle of services added as a group to an account. Packages specify default values for bill layout and billing frequencies, and are used to order subscriptions and accounts. |

**Note:** For a full list of Smile terms, see the Glossary .

## API components

The Smile API consists of two components: the SOAP API and the message bus. Each component is independent, but using both components lets you take advantage of the full range of functionality in the Smile API.

In this guide, *Smile API* refers to both the SOAP API and the message bus, and *SOAP API* refers to the SOAP methods you use to interact with Smile.

### API architecture

The following diagram provides an overview of the API architecture. The bridge component translates Smile's internal messages into a format suitable to publish to the message bus.

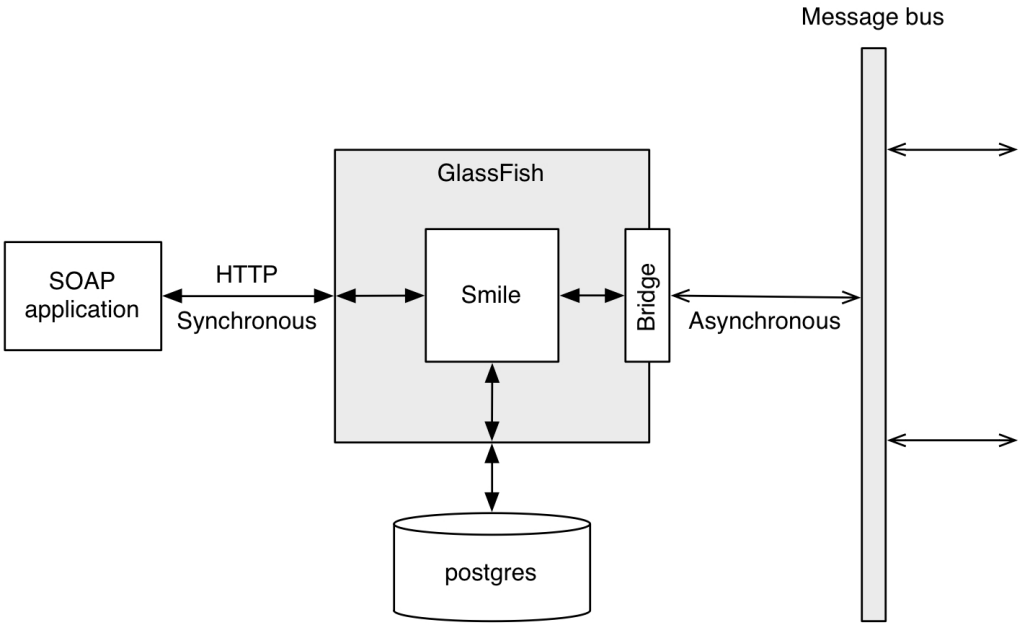



**Figure 1: API architecture**

### The SOAP API

The SOAP API exposes web services that clients use to programmatically interact with Smile. SOAP is a platform- and language-independent XML protocol that permits HTTP communication between applications.

Use the SOAP API to write applications that interact with Smile, for example, applications that get data, place orders or manage subscriptions and accounts. The applications can also listen for messages on the message bus and take further action in response to these messages.

## The message bus

The message bus lets you send messages between two or more clients. Smile publishes Java Message Service (JMS) messages to a message bus, and a listener filters the messages on the bus and retrieves the messages in which they are interested. Message bus communication is asynchronous, and there may be a delay between a client publishing a message and a listener receiving it.

Smile messages have the following components:

- **header**—contains routing and identification information. Smile headers contain a smileMessageType string to help you filter the messages you listen for.
- **body**—contains the data to exchange. Smile messages have a text message body that contains an XML document.

Smile sends messages to the message bus to notify listeners of changes to Smile objects, for example, updated subscriptions, completed orders or allocated transactions. Writing an application that listens for messages lets you synchronise changes in Smile across your network. For example, a listener can listen for transaction messages to update the report system or for provisioning messages that must be forwarded to an upstream provider.

Although Smile emits a message to confirm that a change is complete, the message does not imply that Smile was the application that made the change or that the change originated with a SOAP method. For example, Smile emits a message after a subscription is provisioned, but the provisioning can occur at an upstream provider. Similarly, Smile emits the same message when a subscription is updated by a SOAP method or in Smile.

Although Smile publishes JMS messages, you do not need Java to receive the messages. However, your message broker must be JMS compatible and have a GlassFish resource adapter.

## Example: SOAP methods and the message bus

This example demonstrates how you can write applications that use both the SOAP API and the message bus to perform tasks in Smile. In this example, Client 1 requests a subscription update. Smile updates the subscription and publishes a message to the bus. Client 2 receives the message after a delay and checks that the message contains the most up-to-date information about the subscription.

1. Client 1 updates a subscription, as follows:
   a. Client 1 sends a SOAP message to Smile that contains the subscription `update()` method.
   b. Smile receives the SOAP message and updates the subscription.
   c. Smile sends a SOAP response to Client 1 and the `SubscriptionUpdated` message to the message bus. Both messages contain details of the updated subscription.
2. The Client 2 listener retrieves the `SubscriptionUpdated` message from the message bus.
3. Client 2 verifies that it has the latest version of the updated subscription information, as follows:
   a. Client 2 sends a SOAP message to Smile that contains the subscription `get()` method.
   b. Smile receives the SOAP message and retrieves the subscription details.
   c. Smile sends a SOAP response to Client 2 that contains the current subscription details. Smile does not send a message to the message bus for this transaction because the SOAP method did not change the subscription.

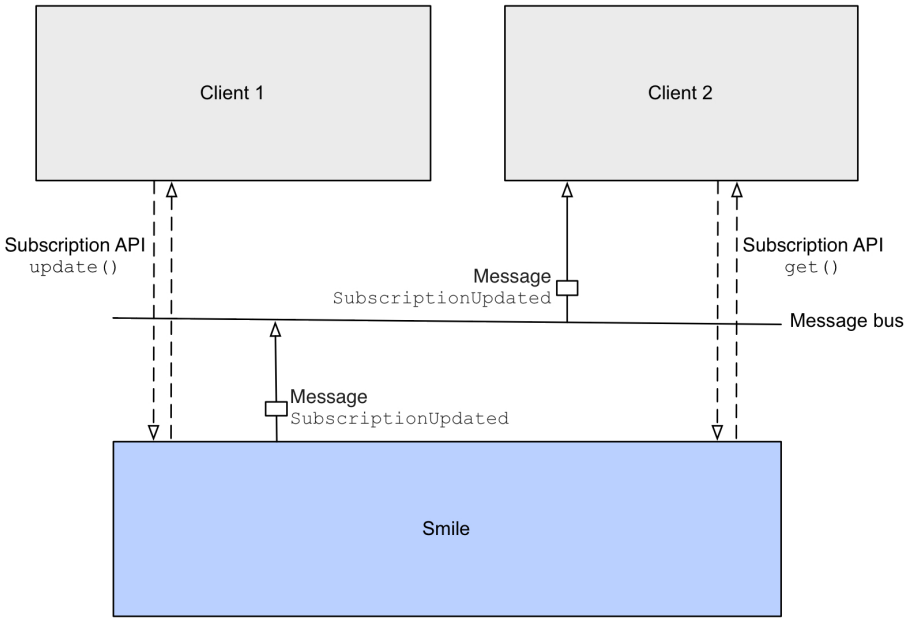



**Figure 2: An example of SOAP methods and the message bus**

### Related concepts

Example: Get a subscription on page 29
Example: Update a subscription on page 32

## Set up the Smile API

To set up the Smile API, you connect separately to the SOAP API and the message bus, as follows:

1. Plan your deployment.
2. Configure SOAP access.
3. Connect to the SOAP API.
4. Connect to the message bus.

You should also test your Smile API implementation after you set it up.

### Plan your deployment

Consider the following points before you set up the Smile API:

- **Platform**—The Smile API is platform-neutral and you do not need Java to access it.
- **Broker**—We recommend that you use Active MQ as your message broker, however you can use any JMS-compatible broker with a GlassFish resource adapter.

Complete the following tasks before you set up the Smile API:

- **Ports**—Ensure port 443 is open for HTTPS communication between the SOAP API and your application. You may also need to open a firewall port for your broker.
- **Security**—Ensure the host SSL certificate is added to the operating system's list of trusted certificates.
- **Disaster recovery**—Plan for disaster recovery scenarios and message broker persistence when planning your Smile deployment.
- **API user**—Obtain a username and password for the API user from Inomial.
- **Software**—Obtain the following software for ActiveMQ:
  - ActiveMQ resource adapter from the ActiveMQ website
  - Smile message bridge from Inomial

  Contact Inomial for software requirements for other brokers.

### Configure SOAP access

A CAT username and password with API access is required. Your application uses these credentials to log in to the API.

This task explains how to configure an account with a CAT access subscription providing API permissions.

1. Create a new account with a CAT Access service subscription or add a CAT Access subscription to an existing account.
2. **View** the CAT Access service information.

   The **Service** information is displayed.
3. On the Account sidebar, click the **Access** sub-menu for the CAT subscription.

   The subscription's **Access** information is displayed.

4. Select the **Allow SOAP API access** checkbox.

   This permission is required to allow this SOAP user to be able to call methods that refer to their own subscription.

   Additional SOAP access options are displayed.

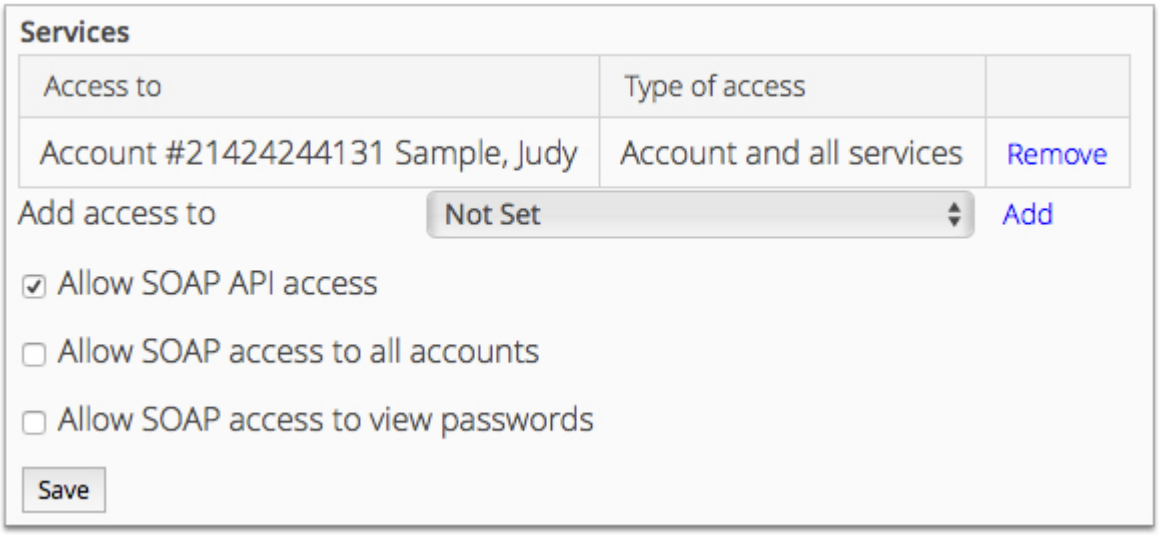
5. Select the **Allow SOAP access to all accounts** checkbox.

   This permission is required to allow this SOAP user to be able to retrieve privileged information or carry out privileged operations relating to other accounts and subscriptions.

6. Click **Save**.

   The SOAP access permissions are saved.

**The Cat Access service Access page**




## Connect to the SOAP API

The SOAP API exposes web services that applications use to interact with Smile.

This task explains how to connect each application to the SOAP API.

1. Review the library you will use to write the API application, or review the functionality of the Smile Java SOAP client.

   The Java client provides an easy way to call methods in the API.

2. Download and review the WSDL and schema for each API endpoint that you want to use.

   Reviewing the WSDL and schema lets you become familiar with the methods and element definitions in the API. The WSDL is located at the following URL, where *endpoint* is the name of the Smile endpoint, for example, *account* or *subscription*.

   **WSDL location:** `http://smile.example.com:port/stage/smile/ws/v2/endpoint?wsdl`

   **Schema location:** Listed in each WSDL.

   Some SOAP libraries use the WSDL to generate stubs, which can also help you write your application.

3. Write a test application using the `echo()` method.

   The API uses HTTP basic access authentication. Ensure you supply the authentication credentials in the correct basic access format for your library.

4. Use the test application to verify the connection to the API.

   You can now connect to the SOAP API.

To use the full range of Smile API functionality, you should now connect to the message bus.

## Connect to the message bus

Connecting to the message bus lets other applications listen for Smile messages, synchronise Smile data across the network and take action as a result of changes in Smile.

This task explains how to connect a broker to the message bus. The broker must be JMS compatible and have a GlassFish resource adapter.

1. Configure the broker to integrate with Smile:

   - Create broker accounts for GlassFish and for your listener. These accounts let Smile and your listener log in to the broker.
   - Configure broker transports for GlassFish and for your listener. These transports manage communication between the broker and application.
   - Configure broker security to suit your environment. Note that ActiveMQ is insecure by default.
   - Consider disaster recovery scenarios for the broker, if you have not already done so.

2. Configure GlassFish to connect to the broker.

   Contact Inomial for information about GlassFish configuration.

3. Make a change in Smile and verify that your listener receives the appropriate message.

   For example, update a subscription and verify that the listener receives the `SubscriptionUpdated` message.

   You have now set up the Smile API.

# Test the Smile API

Smile provides two tools to help you test your SOAP API implementation:

- the Java SOAP client—calls methods in the API
- `echo()`—tests the connection to the API

## The Java SOAP client

The Java SOAP client provides an easy way for you to call methods in the SOAP API. The Java client helps you:

- test the API
- understand how the API works
- review the information the methods return
- troubleshoot the API

The Java client does not support the full range of API functionality, is not intended for use in a production environment and is not bundled with Smile. Contact Inomial to obtain the package for the Java client.

### Run the Java client

Before you run the Java client, ensure Apache Ant and the Java Development Kit (JDK) are installed on the same computer as the client. We recommend that you install Apache Ant 1.7 or later, and the same version of the JDK that you use to run Smile.

The Java client calls SOAP methods and performs basic functions in the API. This task explains how to run the Java client on a Linux computer.

1. Obtain the Java client package from Inomial.

   If you do not use a Linux computer, also obtain the Java client instructions for your operating system from Inomial.
2. Extract the package and navigate to the new `/client.ws.smile.inomial.com` directory, which contains the extracted files.
3. Run the following command to compile the Java client:

   ```
   ant
   ```
4. Run the Java client, using the appropriate syntax for the method you want to call.

   The syntax for each method is documented with the method description.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options] className parameters
```

The Java client is an application that lets you call methods in the SOAP API.

#### *options* syntax

Options provide additional information about how the Java client connects to the API. If you do not specify an option, the Java client uses the default (underlined) value.

Each option has the following syntax: `-DoptionName=value`

**smile.soap.insecure**

**Values:** true, false

**Purpose:** Specifies if the Java client uses insecure communication, that is, ignores invalid and unverified certificates.

**smile.soap.url**

**Values:** http://localhost:8080/smile/ws/v2/

**Purpose:** Specifies the URL and port that the Java client connects to.

**smile.soap.username**

**Values:** soap@inomial.com

**Purpose:** Specifies the username that the Java client uses to connect to the API.

**smile.soap.password**

**Values:** No default.

**Purpose:** Specifies the password that the Java client uses to connect to the API.

#### *className* and *parameters* syntax

The values for each class name and its parameters are documented with the SOAP method that they call. The `/client.ws.smile.inomial.com/examples` directory contains sample XML files for some methods.

#### Example: Use the Java client to raise a ticket

This example calls the `raiseTicket()` method in the CRM endpoint and specifies four options: insecure communication, the URL the Java client connects to, and the username and password the Java client uses to connect to the API.

```
java -cp build/smilewsv2-client.jar -Dsmile.soap.insecure=true
-Dsmile.soap.url="https://localhost/ws/v2"
-Dsmile.soap.username=soap@example.com
-Dsmile.soap.password='pa55w0rd'
com.inomial.smile.client.v2.examples.RaiseTicket
123 2112345678 examples/test-raiseTicket.xml
```

**Note:** For more information about the class name and parameters the Java client uses in this command, see `raiseTicket()`.

### echo()

```
string echo(string value)
```

This method returns any value that is passed to it. Use this method to test an application's connection to the SOAP API. This method is available in the following API endpoints:

- account
- CRM
- order
- services
- subscription

#### Parameters

**value**

**Format:** string

**Purpose:** Contains any arbitrary value.

#### Results

This method returns the value that is passed to it. If no value is returned, there may be a connection problem between the caller and the API.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.EchoTest
```

This class name does not accept any parameters. This class name tests connectivity to the order, service and subscription API endpoints.

#### Related tasks

Connect to the SOAP API on page 22

# Write applications

This section provides information to help you write applications that interact with Smile web services.

The SOAP API uses basic access authentication. Ensure you supply the authentication credentials in the correct basic access format for your library.

## Best practices

Use the following guidelines to help you write applications that integrate with Smile:

- Do not use Apache Axis to write Smile API applications. This restriction occurs because some API elements cause Axis to create classes that have the same name as classes in java.lang.*.
- Use message selectors to select only the messages that are relevant to your application. Using message selectors lets you write less code, because you do not need to filter irrelevant messages, and also improves message bus performance.
- Ensure that the application establishes a durable connection to the message bus. A durable connection ensures that all messages are delivered to your client software, even after downtime.

## WSDL and XSD locations

Reviewing the WSDL and XSD lets you become familiar with the methods and element definitions in the API.

**Table 2: WSDL and XSD locations**

| Name | Location |
|---|---|
| XSD | `http://smile.example.com:port/stage/smile/ws/v2/endpoint?xsd=1` |
| WSDL | `http://`*`smile.example.com:port/stage/`*`smile/ws/v2/`*`endpoint`*`?wsdl` |

**Note:** In this table *endpoint* is the name of the Smile endpoint, for example, *account* or *subscription*.

## Namespaces

The SOAP API uses four namespaces.

**Table 3: Smile namespaces**

| Name | Purpose | URL |
|---|---|---|
| SOAP | Defines the default namespace for the SOAP envelope | `http://schemas.xmlsoap.org/soap/envelope/` |
| Smile data namespace | Defines Smile data types | `http://xml.inomial.com/smile/2.xsd` |
| Smile endpoint namespace | Defines the API methods and parameters | `http://ws.inomial.com/smile.2` |
| SOAP | Defines the default namespace for the SOAP envelope | `http://www.w3.org/2003/05/soap-envelope` |

## SOAP envelopes

The following examples demonstrate the:

- SOAP envelopes you use to call methods in the API
- SOAP envelopes returned for faults
- conventions used to document the SOAP envelopes

### Example: Get a subscription

This example demonstrates SOAP envelopes for the `get()` method in the subscription endpoint. This method searches by USN for a subscription and returns detailed information about a subscription's properties and status.

#### Example: An envelope for a SOAP request

This SOAP envelope requests information for a subscription with USN 21420930. Note the following points:

- the endpoint is defined in the URL of the HTTP request and does not appear in the envelope
- the USN element is in the default namespace

```
<?xml version="1.0" ?>
<Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <get xmlns:="http://xml.inomial.com/smile/2.xsd" xmlns:="http://
ws.inomial.com/smile.2">
      <usn>2142420930</usn>
    </get>
  </Body>
</Envelope>
```

In this guide, the method that corresponds to the SOAP envelope is documented in the following format:

```
Subscription get(2142420930)
```

#### Example: An envelope for a SOAP response

This SOAP envelope contains the information returned for a subscription with USN 21420930.

```
<?xml version='1.0' encoding='UTF-8'?>
<Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <getResponse xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
      <Subscription>
        <USN>2142420930</USN>
        <SID>16</SID>
        <ServiceName>CAT Access</ServiceName>
        <Parent>1000000008</Parent>
        <Properties>
          <Object>
            <String name="abn"/>
            <String name="billAddress"/>
```

```
            <String name="billBuildingName"/>
            <String name="billCountry"/>
            <String name="billFloorNumber"/>
            <String name="billLotNumber"/>
            <String name="billName"/>
            <String name="billNumber"/>
            <String name="billPostCode"/>
            <String name="billPostalDeliveryType"/>
            <String name="billState"/>
            <String name="billSubUnit"/>
            <String name="billSuburb"/>
            <String name="billType"/>
            <String name="comments"/>
            <String name="companyPosition"/>
            <String name="contactFamily"/>
            <String name="contactGiven"/>
            <String name="contactTitle"/>
            <String name="emailAddress"/>
            <String name="fax"/>
            <String name="homePhone"/>
            <String name="jabberAddress"/>
            <String name="mobilePhone"/>
            <String name="name"/>
            <String name="secretAnswer"/>
            <String name="secretQuestion"/>
            <String name="sex"/>
            <String name="streetAddress"/>
            <String name="streetBuildingName"/>
            <String name="streetCountry"/>
            <String name="streetFloorNumber"/>
            <String name="streetLotNumber"/>
            <String name="streetName"/>
            <String name="streetNumber"/>
            <String name="streetPostalDeliveryType"/>
            <String name="streetPostcode"/>
            <String name="streetState"/>
            <String name="streetSubUnit"/>
            <String name="streetSuburb"/>
            <String name="streetType"/>
            <String name="tradingName"/>
            <String name="workPhone"/>
            <String name="username">joe@example.com</String>
            <String name="framedIpAddress"/>
            <String name="framedIpNetmask"/>
            <Integer name="preferredHomeContactTime"/>
            <Integer name="preferredMobileContactTime"/>
            <Integer name="preferredWorkContactTime"/>
            <Boolean name="billPrintName">false</Boolean>
            <Boolean name="streetPrintName">true</Boolean>
            <Timestamp name="dob"/>
          </Object>
        </Properties>
        <ProvisionStatus>Provisioned</ProvisionStatus>
        <SuspendStatus>Active</SuspendStatus>
        <ActivateStatus>Activated</ActivateStatus>
        <EnableStatus>Enabled</EnableStatus>
        <InvoicingCycle>
```

```
          <CycleType>Anniversary</CycleType>
          <CycleDay>24</CycleDay>
        </InvoicingCycle>
        <RatingCycle>
          <CycleType>Anniversary</CycleType>
          <CycleDay>24</CycleDay>
        </RatingCycle>
        <Timezone key="19">Australia/Sydney</Timezone>
        <CostCentre key="1">Example cost centre</CostCentre>
      </Subscription>
    </getResponse>
  </Body>
</Envelope>
```

In this guide, the response contained in the SOAP envelope is documented in the following format:

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142420930</USN>
  <SID>16</SID>
  <ServiceName>CAT Access</ServiceName>
  <Parent> 1000000008</Parent>
  <Properties>
    <Object>
      <String name="username">joe@example.com</String>
      <Boolean name="billPrintName">false</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <CostCentre key="1">Example cost centre</CostCentre>
</Subscription>
```

### Example: Update a subscription

This example demonstrates a SOAP envelope for the `update()` method in the subscription endpoint, which updates subscription information.

#### Example: An envelope for a SOAP request

This SOAP envelope updates the family name, given name, title and email address for a subscription with USN 21420930.

```
<?xml version="1.0" ?>
<Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <update xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
      <usn>2142420930</usn>
      <Subscription>
        <USN>2142420930</USN>
        <Properties>
          <Object>
            <String name="contactFamily">Smith</String>
            <String name="contactGiven">Joe</String>
            <String name="contactTitle">Mr</String>
            <String name="emailAddress">joe@example.com</String>
          </Object>
        </Properties>
      </Subscription>
    </update>
  </Body>
</Envelope>
```

In this guide, the method that corresponds to the SOAP envelope is documented in the following format:

```
Subscription update(2142420930 SubscriptionXML)
```

And the SubscriptionXML parameter is documented in the following format:

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <Properties>
    <Object>
      <String name="contactFamily">Smith</String>
      <String name="contactGiven">Joe</String>
      <String name="contactTitle">Mr</String>
      <String name="emailAddress">joe@example.com</String>
    </Object>
  <Properties>
</Subscription>
```

### Example: Faults

This example demonstrates a SOAP fault envelope returned by the `getOrderForm()` method in the order endpoint.

#### Example: An envelope for a SOAP fault

This SOAP fault is returned when an order form does not exist. Note that your application determines the format in which an error message is displayed to a user.

```
<?xml version='1.0' encoding='UTF-8'?>
<Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
      <faultcode>S:Server</faultcode>
      <faultstring>NoSuchOrderFormException [pkgId=123]. Use
 getOrderForms() to discover available order forms</faultstring>
      <detail>
        <OrderException xmlns:="http://ws.inomial.com/smile.2"
 xmlns:="http://xml.inomial.com/smile/2.xsd">
          <message>NoSuchOrderFormException [pkgId=123]. Use
 getOrderForms() to discover available order forms</message>
        </OrderException>
      </detail>
    </Fault>
  </Body>
</Envelope>
```

In this guide, this fault is documented in the following format:

```
OrderException
```

# Integrate Smile

This section provides information to help you integrate Smile with your existing network.

To synchronise Smile data across the network, write an application to listen for Smile messages and take appropriate action.

To extend or customise Smile functionality, use one or more of the following features:

- custom fields
- custom collections
- accessory data

## Custom fields

Custom fields let you define specialisations within Smile's existing schema. For example, you can define custom fields that let you map fields in different databases, construct workflows or specify display properties.

Using custom fields, you can define additional:

- **fields**—for example, add a custom client ID field to a subscription form
- **metadata**—for example, specify display properties for a field in a web sign-up form

In Smile's schema, custom field information is contained in the `<Object/>` element. To view custom field information, log in to Smile and select **Configuration** from the main Smile drop-down menu. In the **Service, Ordering & Rating** section click **Custom Fields**.

**Note:** For more information about configuring and using custom fields, contact Inomial.

### Existing field names

The following table lists the field names that are predefined in Smile. To avoid unexpected behaviour, do not give a custom field the same name as an existing field.

| | | |
|---|---|---|
| • abn<br>• billAddress<br>• billBuildingName<br>• billCountry<br>• billFloorNumber<br>• billLotNumber<br>• billName<br>• billNumber<br>• billPostalDeliveryType<br>• billPostCode<br>• billPrintName<br>• billState<br>• billSubUnit<br>• billSuburb<br>• billType<br>• billTypeSuffix<br>• comments<br>• companyPosition | • contactFamily<br>• contactGiven<br>• contactTitle<br>• dob<br>• emailAddress<br>• fax<br>• framedIpAddress<br>• framedIpNetmask<br>• homePhone<br>• jabberAddress<br>• mobilePhone<br>• name<br>• preferredHomeContactTime<br>• preferredMobileContactTime<br>• preferredWorkContactTime<br>• secretAnswer<br>• secretQuestion<br>• sex | • streetAddress<br>• streetBuildingName<br>• streetCountry<br>• streetFloorNumber<br>• streetLotNumber<br>• streetName<br>• streetNumber<br>• streetPostalDeliveryType<br>• streetPostcode<br>• streetPrintName<br>• streetState<br>• streetSubUnit<br>• streetSuburb<br>• streetType<br>• streetTypeSuffix<br>• tradingName<br>• username<br>• workPhone |

## Custom collections

Custom collections are an enumeration of values available to a custom field. You can use a custom collection to specify a value for a custom field.

For example, you can define a custom collection of equipment that may be included with a new subscription, or define a custom collection of plans to assign to a workflow.

To view custom collection information, log in to Smile and select **Configuration** from the main Smile drop-down menu. In the **Service, Ordering & Rating** section click **Custom Collections**.

**Note:** For more information about configuring and using custom collections, contact Inomial.

## Accessory data

Accessory data is a complex customisation, defined in code by Inomial, that extends the XML schema of your Smile implementation. The extended schema may appear in subscription, plan and service XML documents, among others, and each schema is tailored to a specific Smile instance.

Accessory data may help you integrate your existing systems with Smile and lets you define more complex specialisations than you can achieve using custom fields.

**Note:** For more information about accessory data and the extended schema, contact Inomial.

## Recommendations after a Smile upgrade

The Smile API is very rich and is continually being extended. After an upgrade to Smile we recommend that you recompile your applications to regenerate any generated code stubs and ensure continued compatibility. Smile API users should undertake testing after an upgrade to confirm that all functions work as intended.

# Chapter 3

# Placing orders

## Order API overview

The order endpoint of the SOAP API lets you order accounts and subscriptions.

Each order specifies the package that Smile uses to create the account or subscription. A package contains one or more services and the configuration information needed to set up the services, such as stationery and dispatch rules for an account, or plan schedules and usage allowances for a subscription. Packages help streamline the order process, because you can order an account and multiple subscriptions at the same time. Packages also let you bundle services that cannot be purchased individually.

Smile emits SmileOrder messages after it completes an order action. These messages help you track the status of orders, and can signal to other applications that an action is required to process the order.

The order WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/order?wsdl`

### Related reference

Order endpoint on page 440
SmileOrder on page 484

## How to use the order API

Packages let you create accounts and subscriptions in Smile. This process explains how to use the order endpoint of the API to order a package.

1. Configure the package for API ordering.

   Packages cannot be ordered in the API by default. You must complete additional steps in Smile before you can use the API to order a package.
2. Retrieve the order form.

   The order form lists the fields that must be included in an API order.
3. Place an order for the package.
4. Approve the order, if the package specifies that approval is required. If an order requires approval, you may need to complete the following steps:
   a. Retrieve order information.

      This lets you display the order to an approver.
   b. Update the order to provide additional information.

      For example, you may need to enter additional information such as a SIM card number or an approver may wish to change the order details. You cannot update an order after it is approved.
   c. Approve the order.

   The order is now complete. Smile automatically approves and completes any orders that do not require approval.

   **Note:** You can also use the Service order approvals report to approve orders in Smile.

To ensure that other applications take necessary actions after an order is complete, write an application that listens for SmileOrder messages.

### Related reference

SmileOrder on page 484

## Configure a package for API ordering

By default, API ordering is not enabled when you create a package. This task explains how to configure a package so you can order it through the API.

1. Log in to Smile.
2. On the **Smile** menu, click **Configuration**.

   The Configuration and Tools page is displayed.
3. Under **Services, Ordering and Rating**, click **Packages**.

   The Packages page is displayed.
4. Click **View** to edit an existing package.

   Information about the package is displayed.
5. Click the **Order Form** tab and add fields to the package order form.

   Your application uses these fields to order the package in the API.

   **Note:** Select the **Use Order Form** checkbox only if you want to display this subset of fields when Smile users order the package.
6. Click the **Plans** tab and add plans to the package.

   A package can contain more than one plan for a service. For example, a broadband package can contain a choice between plans with downloads of 1GB/month or 5GB/month.
7. Publish the package if you have not already done so.

   You cannot order an unpublished package.
8. Enable CAT ordering for the API user, as follows:
   a. Search for the name of the API user.

      This is the name your application uses to connect to the API.
   b. Double-click the CAT entry to view the subscription details.

      The subscription summary for the API user is displayed.
   c. On the Account sidebar, click the **CAT Ordering options** sub-menu for the CAT subscription.
   d. Select the packages you want the API user to order, then click **Add**.

   The API user can now order this package through the API.

## Retrieve order forms

Retrieving order forms lets you get information about the fields in an order form and the plans that you can order in a package. An application uses this information to place an order in the API.

The following methods retrieve order forms:

- `getAvailableOrderForms()` returns information about each package the API user can order. Use this method to list the order forms and plans for each package that an application is authorised to order.
- `getOrderForms()` returns information about the order form and plans for a specific package.

### getAvailableOrderForms()

```
OrderFormList getAvailableOrderForms()
```

This method returns information about the order form and plans in each package the API user is authorised to order. Use this method to list the packages that an application can order in the API.

#### Parameters

This method does not accept any parameters.

#### Results

This method returns an OrderFormList XML document that lists the plans and order forms in each package the application is authorised to order.

There are no messages defined by this method.

#### Example: Attributes for fields in an order form

This example shows a snippet of a returned OrderFormList XML document. This snippet describes the attributes for a field in the order form.

The nested elements in the `<AttributeType>` element specify the format of the field, as follows:

| | |
|---|---|
| **AttributeType** | Specifies the database property for the field type. This attribute is for internal use and you can ignore it. |
| **Name** | Specifies the code name for the field. Your application must use this attribute when ordering a package. |
| **Label** | Specifies the code name label. This attribute is for internal use and you can ignore it. |
| **SmileObjectXMLType** | Specifies the object type for the field, as defined in the Smile XSD. Your application must use this attribute when ordering a package. |

| | |
|---|---|
| **AttributeTypeProperties** | Specifies any objects stored in the field. This attribute maps to the attribute properties field in Smile and lets you attach objects that contain metadata to the order form. Smile does not attach any meaning to objects stored in this field. This attribute lets you store data in Smile, for example, metadata that specifies when a web interface displays the field. |

This example shows the attributes for a required username field that accepts any string. The field does not store any objects.

```
<Attribute>
  <Required>true</Required>
  <Label>Username</Label>
  <Visibility>VisibleAndEditable</Visibility>
  <AttributeType>
    <AttributeType>92</AttributeType>
    <Name>username</Name>
    <Label>Username</Label>
    <SmileObjectXMLType type="String"/>
    <AttributeTypeProperties>
      <Object/>
    </AttributeTypeProperties>
  </AttributeType>
</Attribute>
```

### Example: A returned OrderFormList XML document

This example shows an OrderFormList XML document that contains a single order form. The order form is for a package named ADSL with VOIP Lines, which has a package ID of 192 and contains four plans. The order form contains four fields (username, phone lines, given name and family name).

```
<OrderFormList xmlns="http://xml.inomial.com/smile/2.xsd">
  <OrderForm>
    <Package>192</Package>
    <Name>ADSL With VOIP Lines</Name>
    <OrderFormType>Subscription</OrderFormType>
    <PackagePlans>
      <PackagePlan>
        <PackagePlan>812</PackagePlan>
        <Name>ADSL1 8000/384</Name>
      </PackagePlan>
      <PackagePlan>
        <PackagePlan>89</PackagePlan>
        <Name>ADSL1 256/64</Name>
      </PackagePlan>
      <PackagePlan>
        <PackagePlan>93</PackagePlan>
        <Name>ADSL1 1500/256</Name>
      </PackagePlan>
      <PackagePlan>
        <PackagePlan>95</PackagePlan>
        <Name>ADSL1 512/512</Name>
      </PackagePlan>
    </PackagePlans>
```

```
    <OrderFormDescription>
      <Form>
        <Name>ADSL - Order Form</Name>
        <FormProperties/>
        <Attributes>
          <Attribute>
            <Required>true</Required>
            <Label>Username</Label>
            <Visibility>VisibleAndEditable</Visibility>
            <AttributeType>
              <AttributeType>92</AttributeType>
              <Name>username</Name>
              <Label>Username</Label>
              <SmileObjectXMLType type="String"/>
              <AttributeTypeProperties>
                <Object/>
              </AttributeTypeProperties>
            </AttributeType>
          </Attribute>
          <Attribute>
            <Required>true</Required>
            <Label>Phone Lines</Label>
            <Visibility>VisibleAndEditable</Visibility>
            <AttributeType>
              <AttributeType>91</AttributeType>
              <Name>phoneLines</Name>
              <Label>Phone Lines</Label>
              <SmileObjectXMLType type="Integer"/>
              <AttributeTypeProperties>
                <Object/>
              </AttributeTypeProperties>
            </AttributeType>
          </Attribute>
          <Attribute>
            <Required>true</Required>
            <Label>Contact's Given Name</Label>
            <Visibility>VisibleAndEditable</Visibility>
            <AttributeType>
              <AttributeType>90</AttributeType>
              <Name>contactGiven</Name>
              <Label>Contact's Given Name</Label>
              <SmileObjectXMLType type="String"/>
              <AttributeTypeProperties>
                <Object/>
              </AttributeTypeProperties>
            </AttributeType>
          </Attribute>
          <Attribute>
            <Required>true</Required>
            <Label>Contact's Family Name</Label>
            <Visibility>VisibleAndEditable</Visibility>
            <AttributeType>
              <AttributeType>89</AttributeType>
              <Name>contactFamily</Name>
              <Label>Contact's Family Name</Label>
              <SmileObjectXMLType type="String"/>
              <AttributeTypeProperties>
```

```
                <Object/>
              </AttributeTypeProperties>
            </AttributeType>
          </Attribute>
        </Attributes>
      </Form>
    </OrderFormDescription>
  </OrderForm>
</OrderFormList>
```

### Faults

`OrderException`

This fault is returned when the caller does not have permission to access any order forms.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAvailableOrderForms
```

This class name does not accept any parameters.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getOrderForm()`

```
OrderForm getOrderForm(int packageId)
```

This method returns information about the order form and plans for a specific package. Use this method to get detailed information about a specific order form.

### Parameters

**packageId**

**Format:** integer

**Purpose:** Specifies the package ID. The package ID is located in the **Package ID** field of the **Package** tab when you view the package in Smile, and is also specified in the XML document returned by the `getAvailableOrderForms()` method.

### Results

This method returns an OrderForm XML document that lists the fields in the package order form, and the plans in the package.

There are no messages defined by this method.

### Example: A returned OrderForm XML document

This example shows an OrderForm XML document for a subscription package. The package contains four plans and the order form contains four fields (username, phone lines, given name and family name).

```
<OrderForm xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
```

```
  <Name>ADSL With VOIP Lines</Name>
  <OrderFormType>Subscription</OrderFormType>
  <PackagePlans>
    <PackagePlan>
      <PackagePlan>812</PackagePlan>
      <Name>ADSL1 8000/384</Name>
    </PackagePlan>
    <PackagePlan>
      <PackagePlan>89</PackagePlan>
      <Name>ADSL1 256/64</Name>
    </PackagePlan>
    <PackagePlan>
      <PackagePlan>93</PackagePlan>
      <Name>ADSL1 1500/256</Name>
    </PackagePlan>
    <PackagePlan>
      <PackagePlan>95</PackagePlan>
      <Name>ADSL1 512/512</Name>
    </PackagePlan>
  </PackagePlans>
  <OrderFormDescription>
    <Form>
      <Name>ADSL - Order Form</Name>
      <FormProperties/>
      <Attributes>
        <Attribute>
          <Required>true</Required>
          <Label>Username</Label>
          <Visibility>VisibleAndEditable</Visibility>
          <AttributeType>
            <AttributeType>92</AttributeType>
            <Name>username</Name>
            <Label>Username</Label>
            <SmileObjectXMLType type="String"/>
            <AttributeTypeProperties>
              <Object/>
            </AttributeTypeProperties>
          </AttributeType>
        </Attribute>
        <Attribute>
          <Required>true</Required>
          <Label>Phone Lines</Label>
          <Visibility>VisibleAndEditable</Visibility>
          <AttributeType>
            <AttributeType>91</AttributeType>
            <Name>phoneLines</Name>
            <Label>Phone Lines</Label>
            <SmileObjectXMLType type="Integer"/>
            <AttributeTypeProperties>
              <Object/>
            </AttributeTypeProperties>
          </AttributeType>
        </Attribute>
        <Attribute>
          <Required>true</Required>
          <Label>Contact's Given Name</Label>
          <Visibility>VisibleAndEditable</Visibility>
```

```
          <AttributeType>
            <AttributeType>90</AttributeType>
            <Name>contactGiven</Name>
            <Label>Contact's Given Name</Label>
            <SmileObjectXMLType type="String"/>
            <AttributeTypeProperties>
              <Object/>
            </AttributeTypeProperties>
          </AttributeType>
        </Attribute>
        <Attribute>
          <Required>true</Required>
          <Label>Contact's Family Name</Label>
          <Visibility>VisibleAndEditable</Visibility>
          <AttributeType>
            <AttributeType>89</AttributeType>
            <Name>contactFamily</Name>
            <Label>Contact's Family Name</Label>
            <SmileObjectXMLType type="String"/>
            <AttributeTypeProperties>
              <Object/>
            </AttributeTypeProperties>
          </AttributeType>
        </Attribute>
      </Attributes>
    </Form>
  </OrderFormDescription>
</OrderForm>
```

### Faults

`OrderException`

This fault is returned when the package does not exist or does not have an order form, or the caller does not have permission to order the package.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetOrderForm packageID
```

***packageID***

**Purpose:** Specifies the package ID number.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Related concepts

Example: Faults on page 33

## Place an order

An order specifies the package that Smile uses to create an account or subscription. The following methods place orders:

- `orderAccount()` orders a package that contains an account, and may contain one or more subscriptions.
- `orderSubscription()` orders a package that contains only subscriptions. You cannot use this method to order a package that contains an account.

The `reserveUsns()` method reserves subscription USNs in Smile that can be used for ordering later.

### `orderAccount()`

```
OrderDetails orderAccount(Order order)
```

This method orders a package that contains an account and that may contain one or more subscriptions. The package specifies the configuration information for the account and subscriptions, and also specifies if the order requires approval.

Before you call this method, call `getOrderForm()` to get the package order form.

#### Parameters

**order**

**Format:** Order XML

**Purpose:** Defines an Order XML document that specifies the package to order and contains the completed package order form. If the package contains plans, this document also specifies the plan to order.

#### Example: Order an account

This example orders package 192 with plan 812. The package order form has seven fields (username, given name, family name, email address, company name, number of phone lines and secondary contact). Secondary contact is a compound custom field type, with sub-fields of secondary contact name and secondary contact number.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <String name="emailAddress">joe@example.com</String>
      <String name="companyName">Example Telecom Ltd</String>
      <Integer name="phoneLines">12</Integer>
      <Object name="secondaryContact">
        <String name="secondaryContactName">John Smith</String>
        <String name="secondaryContactNo">+61395505665</String>
```

```
      </Object>
    </Object>
  </OrderForm>
</Order>
```

### Results

This method returns an OrderDetails XML document that contains the status of the order, the order number and information about the ordered account and any ordered subscriptions.

Smile emits the `OrderEntered` message after the order is entered.

### A returned OrderDetails XML document

This example shows an OrderDetails XML document for an account and two subscriptions. The document contains the order number (O12492318), specifies that the order has been approved (RequiresApproval is false), and contains the USN, SID and status information (provisioned, activated and enabled) for the account and subscriptions.

```
<OrderDetails xmlns="http://xml.inomial.com/smile/2.xsd">
  <OrderNumber>O12492318</OrderNumber>
  <Order>
    <Package>192</Package>
    <RequiresApproval>false</RequiresApproval>
    <AccountUSN>1204921324</AccountUSN>
    <PackagePlan>812</PackagePlan>
    <OrderForm>
      <Object>
        <String name="username">+61355501256</String>
        <String name="contactGiven">Joe</String>
        <String name="contactFamily">Smith</String>
        <String name="emailAddress">joe@example.com</String>
        <String name="companyName">Example Telecom Ltd</String>
        <Integer name="phoneLines">12</Integer>
        <Object name="secondaryContact">
          <String name="secondaryContactName">John Smith</String>
          <String name="secondaryContactNo">+61395505665</String>
        </Object>
      </Object>
    </OrderForm>
  </Order>
  <OrderedComponents>
    <Account>
      <USN>1204921324</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
    </Account>
    <Subscription>
      <USN>120398123</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>15</SID>
    </Subscription>
```

```
    <Subscription>
      <USN>120398131</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>1</SID>
    </Subscription>
  </OrderedComponents>
</OrderDetails>
```

### Faults

`OrderException`

This fault is returned when Smile cannot process the order, or the `<Order>` element is not specified or is in the wrong namespace.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.OrderAccount filename
```

***filename***

**Purpose:** Defines the filename of the Order XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `orderSubscription()`

```
OrderDetails orderSubscription(string usn, Order order)
```

This method orders a package that contains one or more subscriptions. The package specifies the configuration information for the subscriptions and also specifies if the order requires approval.

Before you call this method, call `getOrderForm()` to get the package order form. You must create an account before you can order a subscription against the account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to raise the subscription against. This USN must match the USN in the Order XML document.

**order**

**Format:** Order XML

**Purpose:** Defines an Order XML document that specifies the package to order and contains the completed package order form. If the package contains plans, this document also specifies the plan to order.

### Example: Order a subscription

This example orders package 192 with plan 812 against account 1204921324. The order form for the package has five fields (username, given name, family name, number of phone lines and secondary contact). Secondary contact is a compound custom field type, with sub-fields of secondary contact name and secondary contact number.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>true</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
      <Object name="secondaryContact">
        <String name="secondaryContactName">John Smith</String>
        <String name="secondaryContactNo">+61395505665</String>
      </Object>
    </Object>
  </OrderForm>
</Order>
```

### Results

This method returns an OrderDetails XML document that contains the status of the order, the order number and information about the ordered subscriptions.

Smile emits the `OrderEntered` message after the order is entered.

### Example: A returned OrderDetails XML document

This example shows an OrderDetails XML document for two subscriptions. The document contains the order number (O12492318), specifies that the order has been approved (RequiresApproval is false) and contains the USN, SID and status information for the subscriptions.

```
<OrderDetails xmlns="http://xml.inomial.com/smile/2.xsd">
  <OrderNumber>O12492318</OrderNumber>
  <Order>
    <Package>192</Package>
    <RequiresApproval>false</RequiresApproval>
    <AccountUSN>1204921324</AccountUSN>
    <PackagePlan>812</PackagePlan>
    <OrderForm>
      <Object>
        <String name="username">+61355501256</String>
        <String name="contactGiven">Joe</String>
        <String name="contactFamily">Smith</String>
        <Integer name="phoneLines">12</Integer>
        <Object name="secondaryContact">
          <String name="secondaryContactName">John Smith</String>
```

```
          <String name="secondaryContactNo">+61395505665</String>
        </Object>
      </Object>
    </OrderForm>
  </Order>
  <OrderedComponents>
    <Subscription>
      <USN>120398123</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>15</SID>
    </Subscription>
    <Subscription>
      <USN>120398131</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>1</SID>
    </Subscription>
  </OrderedComponents>
</OrderDetails>
```

### Faults

`OrderException`

This fault is returned when Smile cannot process the order, or the `<Order>` element is not specified or is in the wrong namespace.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.OrderSubscription usn filename
```

***usn***

**Purpose:** Specifies the USN of the account to raise the subscription against. This USN must match the USN in the Order XML document.

***filename***

**Purpose:** Defines the filename of the Order XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `reserveUsns()`

```
UsnList reserveUsns(int sid, int count)
```

This method reserves subscription USNs in Smile that can be used for ordering later.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service from where the prefix and suffix would apply to the returned USNs.

**count**

**Format:** integer

**Purpose:** Specifies how many USNs should be returned.

The default maximum is 10,000. The default can be changed by setting the system property com.inomial.smile.api.impl.OrderBean.MaxUsnCount. Cannot be bigger than 2,147,483,647. Negative value is not allowed.

#### Results

This method returns a UsnList XML document that contains a list of the reserved USNs.

There are no messages defined by this method.

#### Example: A returned UsnList XML document

This example shows a UsnList XML document listing ten USNs that have been reserved.

```
<UsnList>
  <Usn>2143826226</Usn>
  <Usn>2143826234</Usn>
  <Usn>2143826242</Usn>
  <Usn>2143826259</Usn>
  <Usn>2143826267</Usn>
  <Usn>2143826275</Usn>
  <Usn>2143826283</Usn>
  <Usn>2143826291</Usn>
  <Usn>2143826309</Usn>
  <Usn>2143826317</Usn>
</UsnList>
```

#### Faults

`NoSuchItemException`

This fault is returned when the service does not exist.

`InvalidRequestException`

This fault is returned when the count exceeds the limit.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ReserveUsns sid count
```

***sid***

**Purpose:** Specifies the service from where the prefix and suffix would apply to the returned USNs.

***count***

**Purpose:** Specifies how many USNs should be returned.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Approve an order

Approving an order certifies that the order form contains all necessary information. After an order is approved, it proceeds to the next stage of the order process, as follows:

- accounts move to completion
- subscriptions move to provisioning, activation and enablement

The `approveOrder()` method approves an order.

### `approveOrder()`

```
void approveOrder(string orderNumber)
```

This method approves an order that is pending approval and moves it to the next stage in the order process.

#### Parameters

**orderNumber**

**Format:** string

**Purpose:** Specifies the number or other ID of the order to approve.

#### Results

This method does not return any output. The order proceeds to the next stage of the order process, that is, accounts move to completion and subscriptions move to provisioning, activation and enablement. If you have configured Smile to do so, approving an order may also raise an initial invoice.

Smile emits the `OrderApproved` message after the order is approved.

#### Faults

`OrderException`

This fault is returned when the order is already approved or cannot be approved.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ApproveOrder orderNumber
```

***orderNumber***

**Purpose:** Specifies the number or other ID of the order to approve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Retrieve order information

Retrieving order information lets you view the status of the order components and the information in the completed order form. It also lets you provide information about the order to other systems, for example, a web interface can use this method to display the order to an approver.

The `getOrder()` method retrieves order information.

### `getOrder()`

```
OrderDetails getOrder(string orderNumber)
```

This method searches for an order by order number, and returns an XML document that contains the completed order form and status information about the order components.

This method retrieves orders in any stage of the order process, including completed orders. However, you may not be able to retrieve orders that:

- are archived or cancelled
- contain a deleted subscription
- were created in a previous version of Smile
- the caller does not have permission to access

#### Parameters

**orderNumber**

**Format:** string

**Purpose:** Specifies the number or other ID of the order to retrieve. Smile returns the `<OrderNumber>` element in the OrderDetails XML document that is returned when you place an order in the API.

#### Results

This method returns an OrderDetails XML document that contains the completed order form, and status information for the accounts and subscriptions in the order.

There are no messages defined by this method.

#### Example: A returned OrderDetails XML document

This example shows an OrderDetails XML document for an account and two subscriptions. The document contains the order number (O12492318), specifies that the order has been approved (RequiresApproval is false), and contains the USN, SID and status information (provisioned, activated and enabled) for the account and subscriptions.

```
<OrderDetails xmlns="http://xml.inomial.com/smile/2.xsd">
  <OrderNumber>O12492318</OrderNumber>
  <Order>
    <Package>192</Package>
    <RequiresApproval>false</RequiresApproval>
    <AccountUSN>1204921324</AccountUSN>
```

```
    <PackagePlan>812</PackagePlan>
    <OrderForm>
      <Object>
        <String name="username">+61355501256</String>
        <String name="contactGiven">Joe</String>
        <String name="contactFamily">Smith</String>
        <String name="emailAddress">joe@example.com</String>
        <String name="companyName">Example Telecom Ltd</String>
        <Integer name="phoneLines">12</Integer>
      </Object>
    </OrderForm>
  </Order>
  <OrderedComponents>
    <Account>
      <USN>1204921324</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
    </Account>
    <Subscription>
      <USN>120398123</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>15</SID>
    </Subscription>
    <Subscription>
      <USN>120398131</USN>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <SID>1</SID>
    </Subscription>
  </OrderedComponents>
</OrderDetails>
```

### Faults

`OrderException`

This fault is returned when the order does not exist or the caller does not have permission to access the order.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetOrder orderNumber
```

***orderNumber***

**Purpose:** Specifies the number or other ID of the order to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Update an order

Updating an order lets you change information before an order is approved. For example, an approver can change the plan for an account or an operator can enter a SIM card number for a subscription. You cannot update an approved order.

The `updateOrder()` method updates an order.

### `updateOrder()`

```
void updateOrder(string orderNumber, Order order)
```

This method updates an unapproved order. Use this method if an operator or external system must provide additional information to process an order.

You cannot update an approved, complete or cancelled order.

#### Parameters

**orderNumber**

**Format:** string

**Purpose:** Specifies the number or other ID of the order to update.

**order**

**Format:** Order XML

**Purpose:** Defines an Order XML document that specifies the order data to update.

#### Example: Update an order

This example changes the ordered plan and billing address. Note that the Order XML document contains only the property to update.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <PackagePlan>8</PackagePlan>
  <OrderForm>
    <Object>
      <String name="billAddress">60 The Esplanade</String>
      <String name="billSuburb">Melbourne</String>
      <String name="billState">VIC</String>
      <String name="billPostCode">3000</String>
      <String name="workPhone">+61355501256</String>
      <String name="emailAddress">demo@example.com</String>
    </Object>
  </OrderForm>
</Order>
```

#### Results

This method does not return any output. There are no messages defined by this method.

### Faults

`OrderException`

This fault is returned when the order does not exist, you cannot update a specified field, the caller does not have permission to update the order or the order is approved, complete or cancelled. This fault is also returned when the `<Order>` element is not specified or is in the wrong namespace.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateOrder orderNumber filename
```

***orderNumber***

**Purpose:** Specifies the number or other ID of the order to update.

***filename***

**Purpose:** Defines the filename of the Order XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Managing order groups

An order group is a groups of orders that are all placed at the same time, so that if one order fails then no orders will be placed. Only one initial invoice is raised containing the charges for all of the orders. You can also generate a quote for the initial invoice of an order group.

The following methods manage order groups:

- `createOrderGroup()` creates a new order group with the initial order provided.
- `addOrderToGroup()` adds an order to an order group.
- `placeOrderGroup()` submits all orders in a order group, and may pay the initial invoice.
- `getOrderGroup()` returns an order group.
- `cancelOrderGroup()` permanently cancels all orders in an order group.
- `cancelOrder()` cancels an order and removes it from the group.
- `getOrderGroupQuote()` retrieves a quote for an order group.

### `createOrderGroup()`

```
OrderGroup createOrderGroup(Order initialOrder)
```

This method creates a new order group with the initial order provided. The order will not be placed until `placeOrderGroup()` is called.

#### Parameters

**initialOrder**

**Format:** Order XML

**Purpose:** Defines an Order XML document that specifies the initial order to create the order group. If you wish to order a new account, this initial order must be for the account.

#### Example: An Order XML document

This example shows an Order XML document that will order Package 258 to create a new account for Joe Smith. The account will not be created until the order group is placed.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>258</Package>
  <OrderForm>
    <Object>
      <String name="account.name">Joe Smith</String>
      <String name="email.address">joe@example.com</String>
    </Object>
  </OrderForm>
</Order>
```

#### Results

This method returns an OrderGroup XML document containing information about the order group.

There are no messages defined by this method.

### Example: A returned OrderGroup XML document

This example shows an OrderGroup XML document containing an order group with a new order group number, a new account USN and the details of the order.

```
<OrderGroup xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <OrderGroupNumber>105090</OrderGroupNumber>
  <AccountUSN>2142472105</AccountUSN>
  <Orders>
    <Order>
      <OrderNumber>105090</OrderNumber>
      <Order>
        <Package>4740</Package>
        <RequiresApproval>true</RequiresApproval>
        <AccountUSN>2142472105</AccountUSN>
        <OrderForm>
          <Object>
            <String name="email.address">joe@example.com</String>
            <String name="account.name">Joe Smith</String>
          </Object>
        </OrderForm>
      </Order>
      <OrderedComponents>
        <Account>
          <USN>2142472105</USN>
          <ProvisionStatus>NotProvisioned</ProvisionStatus>
          <SuspendStatus>Active</SuspendStatus>
          <ActivateStatus>NotActivated</ActivateStatus>
          <EnableStatus>NotEnabled</EnableStatus>
        </Account>
      </OrderedComponents>
    </Order>
  </Orders>
</OrderGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when one of the references in the order group, for example, package, account or package plan, does not exist.

`OrderException`

This fault is returned when there is a problem with the order provided.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateOrderGroup filename
```

***filename***

**Purpose:** Defines the filename of the Order XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### addOrderToGroup()

```
OrderGroup addOrderToGroup(string orderGroupId, Order order)
```

This method adds the order specified to the order group.

#### Parameters

**orderGroupId**

**Format:** string

**Purpose:** Specifies the ID of the order group to which to add the order.

**order**

**Format:** Order XML

**Purpose:** Defines an Order XML document that specifies the order details to add to the order group.

#### Example: An Order XML document

This example shows an Order XML document that will add an ADSL package (61) on "Test Plan" (36) to the order group.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>61</Package>
  <PackagePlan>36</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">joe@adsl.example.com</String>
    </Object>
  </OrderForm>
</Order>
```

#### Results

This method returns an OrderGroup XML document that contains information about the order group.

There are no messages defined by this method.

#### Example: A returned OrderGroup XML document

This example shows an OrderGroup XML document.

```
<OrderGroup xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <OrderGroupNumber>105090</OrderGroupNumber>
  <AccountUSN>2142472105</AccountUSN>
  <Orders>
    <Order>
      <OrderNumber>105090</OrderNumber>
      <Order>
        <Package>4740</Package>
```

```
        <RequiresApproval>true</RequiresApproval>
        <AccountUSN>2142472105</AccountUSN>
        <OrderForm>
          <Object>
            <String name="email.address">joe@example.com</String>
            <String name="account.name">Joe Smith</String>
          </Object>
        </OrderForm>
      </Order>
      <OrderedComponents>
        <Account>
          <USN>2142472105</USN>
          <ProvisionStatus>NotProvisioned</ProvisionStatus>
          <SuspendStatus>Active</SuspendStatus>
          <ActivateStatus>NotActivated</ActivateStatus>
          <EnableStatus>NotEnabled</EnableStatus>
        </Account>
      </OrderedComponents>
    </Order>
    <Order>
      <OrderNumber>105110</OrderNumber>
      <Order>
        <Package>61</Package>
        <RequiresApproval>true</RequiresApproval>
        <AccountUSN>2142472105</AccountUSN>
        <PackagePlan>36</PackagePlan>
        <OrderForm>
          <Object>
            <String name="username">jie@adsl.example.com</String>
            <String name="account.name"></String>
          </Object>
        </OrderForm>
      </Order>
      <OrderedComponents>
        <Subscription>
          <USN>2142472303</USN>
          <ProvisionStatus>NotProvisioned</ProvisionStatus>
          <SuspendStatus>Active</SuspendStatus>
          <ActivateStatus>NotActivated</ActivateStatus>
          <EnableStatus>NotEnabled</EnableStatus>
          <SID>47</SID>
        </Subscription>
      </OrderedComponents>
    </Order>
  </Orders>
</OrderGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when one of the references in the order group, for example, package, account or package plan, does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddOrderToGroup orderGroupId
 filename
```

***orderGroupId***

**Purpose:** Specifies the ID of the order group to which to add the order.

***filename***

**Purpose:** Defines the filename of the Order XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `placeOrderGroup()`

```
OrderPlacementResponse placeOrderGroup(string orderGroupId,
 PaymentRequest paymentDetails)
```

This method submits all the orders in the order group, and may pay the initial invoice.

### Parameters

**orderGroupId**

**Format:** string

**Purpose:** Specifies the ID of the order group to submit for ordering.

**paymentDetails**

**Format:** PaymentRequest XML document

**Purpose:** (Optional) Defines a PaymentDetails XML document that specifies the payment details to pay the initial invoice. You can use `getOrderGroupQuote()` to calculate how much needs to be paid.

### Example: Request a payment

This example shows a request for payment.

```
<PaymentRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <PaymentDetails>
    <PaymentType>Visa</PaymentType>
    <AccountName>Joe Smith</AccountName>
    <AccountNumber>2142472105</AccountNumber>
    <ExpiryDate>2012-12</ExpiryDate>
    <CVV>123</CVV>
    <PaymentAmount>$47.40</PaymentAmount>
    <IncludesSurchargeAmount>0.0</IncludesSurchargeAmount>
    <Properties>
      <Object></Object>
    </Properties>
  </PaymentDetails>
</PaymentRequest>
```

### Results

This method returns an OrderPlacementResponse XML document containing the details of the order group and any payment that was made.

Smile emits the `OrderEntered` message after the orders are entered.

### Example: A returned OrderPlacementResponse XML document

This example shows an OrderPlacementResponse XML document for an order that was submitted, but payment was declined.

```
<OrderPlacementResponse xmlns="http://xml.inomial.com/smile/2.xsd"
  xmlns="http://ws.inomial.com/smile.2">
  <OrderGroup>
    <OrderGroupNumber>105090</OrderGroupNumber>
    <AccountUSN>2142472105</AccountUSN>
    <Orders>
      <Order>
        <OrderNumber>105090</OrderNumber>
        <Order>
          <Package>4740</Package>
          <RequiresApproval>false</RequiresApproval>
          <AccountUSN>2142472105</AccountUSN>
          <OrderForm>
            <Object>
              <String name="email.address">joe@example.com</String>
              <String name="account.name">Joe Smith</String>
            </Object>
          </OrderForm>
        </Order>
        <OrderedComponents>
          <Account>
            <USN>2142472105</USN>
            <ProvisionStatus>Provisioned</ProvisionStatus>
            <SuspendStatus>Active</SuspendStatus>
            <ActivateStatus>Activated</ActivateStatus>
            <EnableStatus>Enabled</EnableStatus>
            <InitialInvoice key="61419800">Tx 61419800 (14/01/15)
 $47.40</InitialInvoice>
          </Account>
        </OrderedComponents>
      </Order>
      <Order>
        <OrderNumber>105110</OrderNumber>
        <Order>
          <Package>61</Package>
          <RequiresApproval>false</RequiresApproval>
          <AccountUSN>2142472105</AccountUSN>
          <PackagePlan>36</PackagePlan>
          <OrderForm>
            <Object>
              <String name="username">amy@adsl.example.com</String>
            </Object>
          </OrderForm>
        </Order>
```

```
        <OrderedComponents>
          <Subscription>
            <USN>2142472303</USN>
            <ProvisionStatus>Provisioned</ProvisionStatus>
            <SuspendStatus>Active</SuspendStatus>
            <ActivateStatus>Activated</ActivateStatus>
            <EnableStatus>Enabled</EnableStatus>
            <InitialInvoice key="61419800">Tx 61419800 (14/01/15)
 $47.40</InitialInvoice>
            <SID>47</SID>
          </Subscription>
        </OrderedComponents>
      </Order>
    </Orders>
  </OrderGroup>
  <PaymentResponse>
    <Payment>
      <PaymentNumber>61420006</PaymentNumber>
      <PaymentType>
        <Name>Visa</Name>
        <PaymentTypeCode>visa</PaymentTypeCode>
      </PaymentType>
      <Account>2142472105</Account>
      <Amount currency="AUD" formatted="$47.40">47.40</Amount>
      <Surcharge>
        <Amount currency="AUD" formatted="$0.00">0</Amount>
        <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
      </Surcharge>
      <Status>Declined</Status>
    </Payment>
  </PaymentResponse>
</OrderPlacementResponse>
```

### Faults

`NoSuchItemException`

This fault is returned when payment type or the order group does not exist.

`OrderException`

This fault is returned when the order fails.

`InvalidRequestException`

This fault is returned when the order group has already been placed or the payment details are not appropriate for the order group. For example, the amount is incorrect.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.PlaceOrderGroup orderGroupId
 [filename]
```

***orderGroupId***

**Purpose:** Specifies the ID of the order group to submit for ordering.

***filename***

**Purpose:** (Optional) Defines the filename of the PaymentRequestType XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getOrderGroup()`

```
OrderGroup getOrderGroup(string orderGroupId)
```

This method returns the order group.

#### Parameters

**orderGroupId**

**Format:** string

**Purpose:** Specifies the number of the order group.

#### Results

This method returns an OrderGroup XML document containing details of the order group.

There are no messages defined by this method.

#### Example: A returned OrderGroup XML document

This example shows an OrderGroup XML document.

```
<OrderGroup xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <OrderGroupNumber>105090</OrderGroupNumber>
  <AccountUSN>2142472105</AccountUSN>
  <Orders>
    <Order>
      <OrderNumber>105090</OrderNumber>
      <Order>
        <Package>4740</Package>
        <RequiresApproval>true</RequiresApproval>
        <AccountUSN>2142472105</AccountUSN>
        <OrderForm>
          <Object>
            <String name="email.address">joe@example.com</String>
            <String name="account.name">Joe Smith</String>
          </Object>
        </OrderForm>
      </Order>
      <OrderedComponents>
        <Account>
          <USN>2142472105</USN>
          <ProvisionStatus>NotProvisioned</ProvisionStatus>
          <SuspendStatus>Active</SuspendStatus>
          <ActivateStatus>NotActivated</ActivateStatus>
          <EnableStatus>NotEnabled</EnableStatus>
        </Account>
      </OrderedComponents>
    </Order>
```

```
    <Order>
      <OrderNumber>105110</OrderNumber>
      <Order>
        <Package>61</Package>
        <RequiresApproval>true</RequiresApproval>
        <AccountUSN>2142472105</AccountUSN>
        <PackagePlan>36</PackagePlan>
        <OrderForm>
          <Object>
            <String name="username">jie@adsl.example.com</String>
            <String name="account.name"></String>
          </Object>
        </OrderForm>
      </Order>
      <OrderedComponents>
        <Subscription>
          <USN>2142472303</USN>
          <ProvisionStatus>NotProvisioned</ProvisionStatus>
          <SuspendStatus>Active</SuspendStatus>
          <ActivateStatus>NotActivated</ActivateStatus>
          <EnableStatus>NotEnabled</EnableStatus>
          <SID>47</SID>
        </Subscription>
      </OrderedComponents>
    </Order>
  </Orders>
</OrderGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when the order group ID does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetOrderGroup orderGroupId
```

***orderGroupId***

**Purpose:** Specifies the number of the order group.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `cancelOrderGroup()`

```
void cancelOrderGroup(string orderGroupNumber)
```

This method permanently cancels all orders in an order group.

#### Parameters

**orderGroupNumber**

**Format:** string

**Purpose:** Specifies the order group ID to cancel.

#### Results

This method does not return a result for a successful cancellation. An exception will be generated if the cancellation attempt fails.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when the order group ID does not exist.

`OrderException`

This fault is returned when an error occurs trying to cancel the orders.

`InvalidRequestException`

This fault is returned when the order group cannot be cancelled because it has already been placed or cancelled.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CancelOrderGroup orderGroupNumber
```

***orderGroupNumber***

**Purpose:** Specifies the order group ID to cancel.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `cancelOrder()`

```
void cancelOrder(string orderNumber)
```

This method cancels the given order and removes it from the group.

#### Parameters

**orderNumber**

**Format:** string

**Purpose:** Specifies the number of the order to cancel.

#### Results

This method does not return a result for a successful cancellation. An exception will be generated if the cancellation attempt fails.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when the order group ID does not exist.

`OrderException`

This fault is returned when an error occurs trying to cancel the order.

`InvalidRequestException`

This fault is returned when the order cannot be cancelled because it has already been placed or cancelled.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CancelOrder orderNumber
```

***orderNumber***

**Purpose:** Specifies the number of the order to cancel.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getOrderGroupQuote()

```
ItemisedTransactionDetail getOrderGroupQuote(string orderGroupId)
```

This method retrieves a quote for the given order group.

#### Parameters

**orderGroupId**

**Format:** string

**Purpose:** Specifies the ID of the order group to quote.

#### Results

This method returns an ItemisedTransactionDetail XML document containing details of the quote.

There are no messages defined by this method.

#### Example: A returned ItemisedTransactionDetail XML document

This example shows an ItemisedTransactionDetail XML document for an order group with an ADSL subscription on a $30/month plan and no set up charge.

```
<ItemisedTransactionDetail xmlns="http://xml.inomial.com/
smile/2.xsd">
  <version>2.0</version>
  <transactionType>Quote</transactionType>
  <company>1</company>
  <usn>2142472105</usn>
  <transactionNumber>61419503</transactionNumber>
  <currency>AUD</currency>
  <amount>47.40</amount>
  <accountType>1</accountType>
  <gstAmount>4.31</gstAmount>
  <salesReportCategory>invoice</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$47.40">47.40</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$4.31">4.31</
formattedGstAmount>
  <unallocatedAmount>47.40</unallocatedAmount>
  <glCode>invoice</glCode>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$47.40">47.40</formattedUnallocatedAmount>
  <transactionDate>2015-01-14+11:00</transactionDate>
  <openDate>2015-01-14+11:00</openDate>
  <quoteNumber>61419503</quoteNumber>
  <createdByUsn>2142423066</createdByUsn>
  <transactionItem>
    <usn>2142472303</usn>
    <lineNumber>0</lineNumber>
    <itemCode>031640</itemCode>
    <chargeFrom>2015-01-14+11:00</chargeFrom>
    <chargeTo>2015-01-31+11:00</chargeTo>
    <description>ADSL recurring charge</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
```

```
    <amount>15.84</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0</discount>
    <count>0</count>
    <gstAmount>1.58</gstAmount>
    <salesReportCategory>unknown</salesReportCategory>
    <glCode>500</glCode>
  </transactionItem>
  <transactionItem>
    <usn>2142472303</usn>
    <lineNumber>1</lineNumber>
    <itemCode>031640</itemCode>
    <chargeFrom>2015-02-01+11:00</chargeFrom>
    <chargeTo>2015-02-28+11:00</chargeTo>
    <description>ADSL recurring charge</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>27.27</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0</discount>
    <count>0</count>
    <gstAmount>2.73</gstAmount>
    <salesReportCategory>unknown</salesReportCategory>
    <glCode>500</glCode>
  </transactionItem>
  <transactionItem>
    <usn>2142472303</usn>
    <lineNumber>2</lineNumber>
    <itemCode>round</itemCode>
    <chargeFrom>2015-01-14+11:00</chargeFrom>
    <chargeTo>2015-01-14+11:00</chargeTo>
    <description>Rounding adjustment</description>
    <quantity>1</quantity>
    <currency>AUD</currency>
    <amount>-0.02</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0</discount>
    <count>0</count>
    <gstAmount>0</gstAmount>
    <salesReportCategory>round</salesReportCategory>
    <glCode>200</glCode>
  </transactionItem>
  <transactionItem>
    <usn>2142472303</usn>
    <lineNumber>3</lineNumber>
    <itemCode>gst</itemCode>
    <chargeFrom>2015-01-14+11:00</chargeFrom>
    <chargeTo>2015-01-14+11:00</chargeTo>
    <description>Includes 10% GST</description>
    <quantity>1</quantity>
    <currency>AUD</currency>
    <amount>4.31</amount>
    <isGst>true</isGst>
    <chargeGst>false</chargeGst>
```

```
    <discount>0</discount>
    <count>0</count>
    <gstAmount>0.00</gstAmount>
    <salesReportCategory>gst</salesReportCategory>
  </transactionItem>
</ItemisedTransactionDetail>
```

### Faults

`NoSuchItemException`

This fault is returned when the order group ID does not exist.

`OrderException`

This fault is returned when an error occurs trying to quote the order.

`InvalidRequestException`

This fault is returned when the order cannot be quoted because it has already been placed or cancelled.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetOrderGroupQuote orderGroupId
```

***orderGroupId***

**Purpose:** Specifies the ID of the order group to quote.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Synchronise order information

Smile sends a message to the message bus when it completes each stage of the order process. Listening for these messages helps other applications track the status of orders, and can signal to other applications that an action is required to process the order.

Smile emits SmileOrder messages as an account or subscription moves through the fulfillment process. Smile emits additional SmileSubscription messages when a subscription is provisioned, enabled and activated.

Smile emits SmileOrder messages when:

- `OrderApproved`—an order is approved and proceeds to the next stage of the order process.
- `OrderCancelled`—an order is cancelled.
- `OrderComplete`—an order is complete and no further work will be done on the order.
- `OrderEntered`—an order is entered and proceeds to the approval stage of the order process.
- `OrderRequiresApproval`—an order requires approval before it can proceed to the next stage of the order process.

### Message sequence—Account orders

The following diagram shows you the sequence of the SmileOrder messages that Smile emits when you order an account. Smile automatically approves any orders that do not require approval.

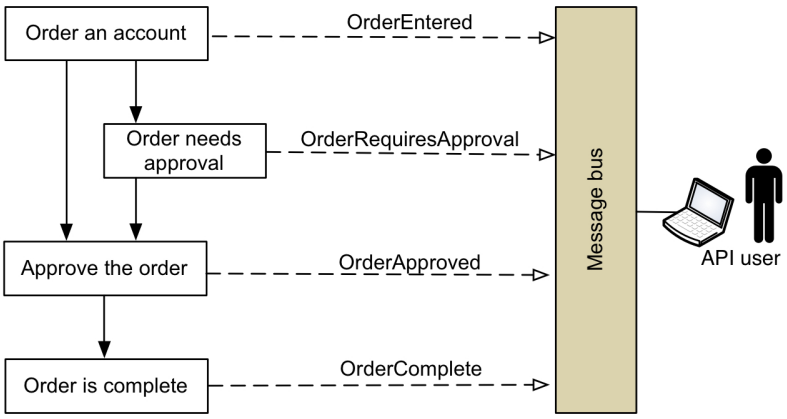



**Figure 3: Messages emitted during the account order process**

Related reference

SmileOrder on page 484

## Message sequence—Subscription orders

The following diagram shows you the sequence of the SmileOrder and SmileSubscription messages that Smile emits when you order a subscription. Smile automatically approves any orders that do not require approval.

You can configure the sequence in which the provision, enable and activate tasks are performed to best suit your business requirements.

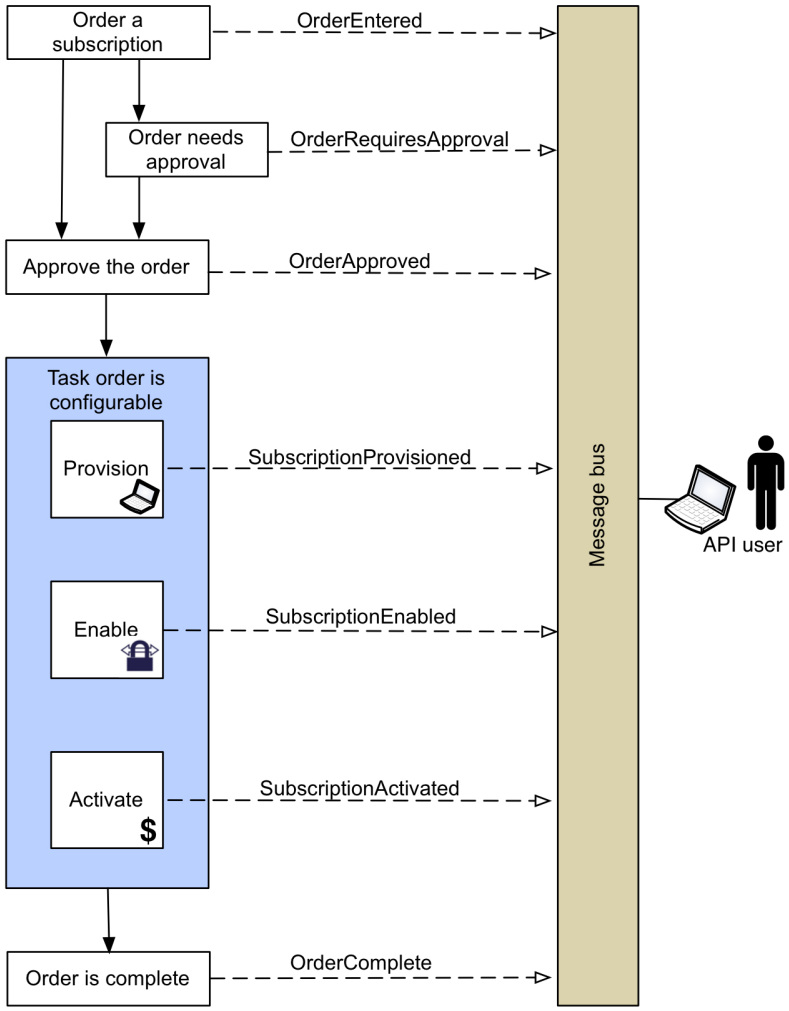



**Figure 4: Messages emitted during the subscription order process**

### Related reference

SmileOrder on page 484
SmileSubscription on page 489

## Message sequence—Provisioning

The following diagram shows you the sequence of SmileSubscription messages that Smile emits when it provisions a subscription, if configured to do so.

The only provisioning message that Smile automatically emits is `SubscriptionProvisioned`. Depending on your provisioning process, Smile may also emit other messages in SmileSubscription. Smile may emit the `SubscriptionProvisionResponse` message multiple times with provisioning updates.

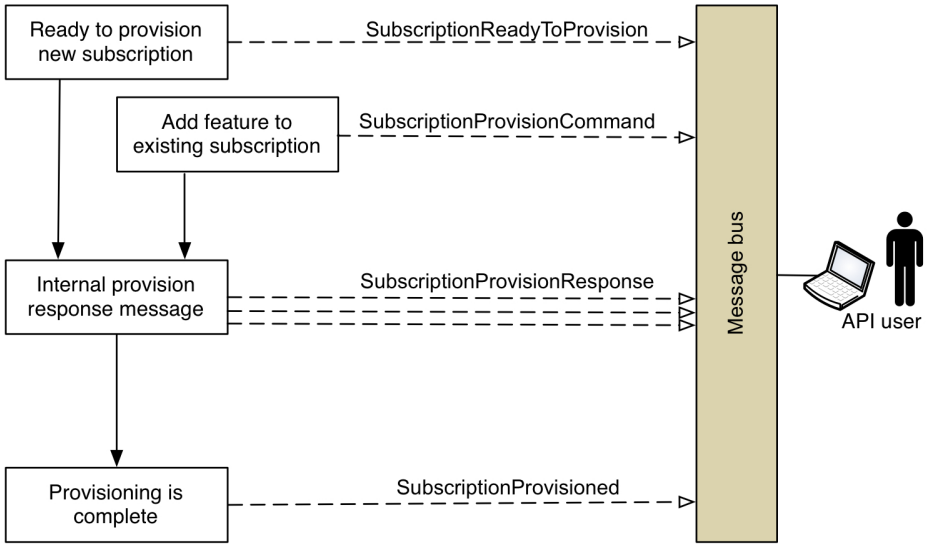



**Figure 5: Messages emitted during the subscription provisioning process**

### Related reference

SmileSubscription on page 489

# Chapter 4

# Managing accounts

## Account API overview

An account is a customer, company or other debtor in the accounts receivable system to whom you provide one or more subscriptions. Actions you can perform on an account include:

- managing contact details
- raising an invoice against the account
- configuring stationery and dispatch settings
- associating tickets and workflows
- assigning the account or a subscription to a cost centre

You configure accounts in Smile. However, you can use the account endpoint of the SOAP API to automate the following account tasks:

- ordering an account
- retrieving account information
- updating an account
- retrieving transaction information
- downloading activity statements
- raising, adding and reversing charges
- compiling potential future charges on an account
- adding and releasing bonds
- creating, updating and deleting invoice groups
- creating and updating purchase orders

The account WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/account?wsdl`

The invoice grouping WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/account/invoicegrouping?wsdl`

### Related reference

Account endpoint on page 419
Invoice grouping endpoint on page 436

## Account structure

Before you use the SOAP API to manage accounts, it is important to understand how changes to an account affect the account's subscriptions.

An account is the node of a tree structure. An account's subscriptions are children of the account, and these subscriptions can have child subscriptions in turn. If you change an account or subscription, the change also applies to the children of the account or subscription.

Examples of account properties that are inherited by subscriptions include:

- discounts
- cost centres
- stationery and dispatch settings
- billing frequency

### Example: A simple tree structure

This example shows you a simple tree structure for an account and subscriptions. Many Smile accounts have this simple structure.

In the following diagram, Joe Smith has subscriptions for mobile, ADSL and CAT services. The mobile subscription is entitled to 5% discount on usage charges.

All subscriptions inherit their contact details, stationery and billing frequency from the account. However, the discount on the mobile subscription is not inherited from the account, so the discount applies only to the subscription.

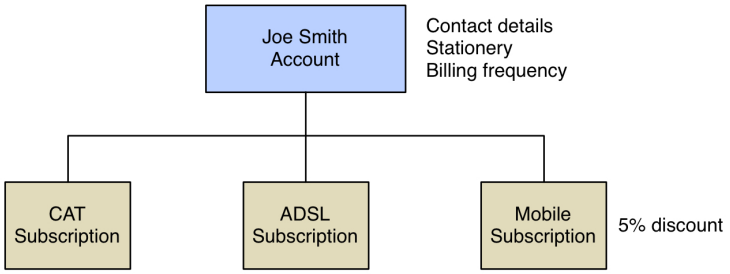



**Figure 6: A simple tree structure**

### Example: A complex tree structure

This example shows you a complex tree structure for an account, subscription and four child subscriptions.

In the following diagram, Company ABC has a data service subscription with a download limit of 1TB/month. This subscription has four ADSL child subscriptions for the company's remote offices; the child subscriptions each have a download limit of 5GB/month.

All subscriptions inherit their stationery and billing frequency from the account. If a remote office downloads more than 5GB in a month, that subscription is shaped but the other child subscriptions are unaffected. However, if the data service subscription downloads more than 1TB in a month, that subscription and all ADSL child subscriptions are shaped.

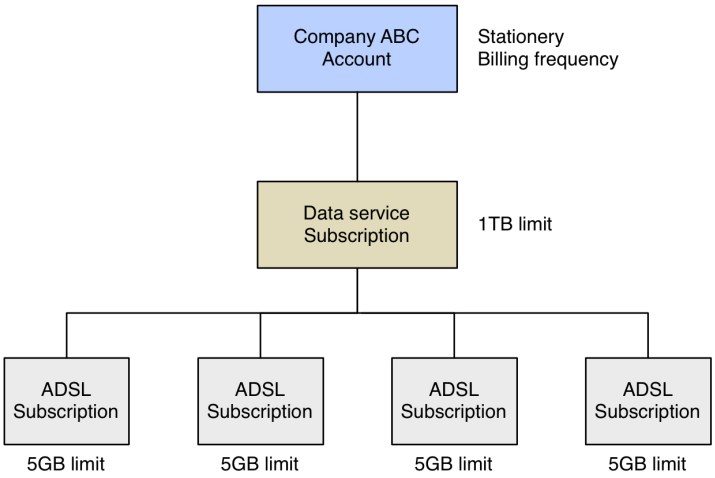



**Figure 7: A complex tree structure**

## Order an account

You use methods in the order endpoint of the SOAP API to order an account; there are no methods in the account endpoint that affect the order process.

### Related concepts

Place an order on page 46

## Retrieve account information

Retrieving account information lets you view the current state of an account and its associated subscriptions.

The following methods retrieve account information:

- `get()` searches by USN for an account and returns information about the account and its subscriptions.
- `getAccountInfo()` searches by USN for an account and returns information about the account.
- `findAccount()` searches by custom field value for an account and returns information about the account and its subscriptions.
- `findAccounts()` searches by custom field for non-closed accounts which are not subscriptions and returns information about any matches.
- `getAccountSummary()` searches by USN for an account and returns information about the account balance, treatment level, aged balances and the number of active subscriptions on the account.

### `get()`

```
Account get(string usn)
```

This method searches by USN for an account and returns an XML document with detailed information about the account and its associated subscriptions.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to retrieve.

#### Results

This method returns an Account XML document that contains detailed information about the account and its associated subscriptions.

There are no messages defined by this method.

#### Example: A returned Account XML document

This example shows an Account XML document. The document contains some account properties (for example, billName and emailAddress) and the subscriptions linked to the account.

**Note:** Smile uses the `key` attribute on the `<Timezone>`, `<AccountTerms>` and `<Costcentre>` elements as an internal identifier.

```
<Account xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>129512142</USN>
  <Invoicee>
    <InvoiceeUsn>2142421482</InvoiceeUsn>
  </Invoicee>
  <InvoicingCycle>
```

```
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <AccountTerms key="1">Default terms</AccountTerms>
  <CostCentre key="105">Example cost centre</CostCentre>
  <Properties>
    <Object>
      <String name="billName">Smith</String>
      <String name="billNumber">60</String>
      <String name="billPostCode">3000</String>
      <String name="billState">VIC</String>
      <String name="billSuburb">Melbourne</String>
      <String name="billType">ST</String>
      <String name="emailAddress">demo@example.com</String>
      <String name="name">Example Company Pty Ltd</String>
      <String name="workPhone">+61355501256</String>
    </Object>
  </Properties>
  <Subscriptions>
    <Subscription>
      <USN>1249871232</USN>
      <SID>17</SID>
      <ServiceName>ADSL</ServiceName>
      <Parent>129512142</Parent>
      <Properties>
        <Object>
          <String name="username">+61355501256</String>
        </Object>
      </Properties>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <SuspendStatus>Active</SuspendStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <InvoicingCycle>
        <CycleType>Anniversary</CycleType>
        <CycleDay>20</CycleDay>
      </InvoicingCycle>
      <RatingCycle>
        <CycleType>Anniversary</CycleType>
        <CycleDay>20</CycleDay>
      </RatingCycle>
      <Timezone key="19">Australia/Sydney</Timezone>
    </Subscription>
  </Subscriptions>
</Account>
```

### Faults

`AccountException`

This fault is returned when the account does not exist or cannot be retrieved, or when the caller does not have permission to access the account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAccount usn
```

***usn***

**Purpose:** Specifies the USN of the account to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAccountInfo()`

```
AccountInfo getAccountInfo(string usn)
```

This method searches by USN for an account and returns an XML document with detailed information about the account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to retrieve.

### Results

This method returns an Account XML document that contains detailed information about the account.

There are no messages defined by this method.

### Example: A returned Account XML document

This example shows an Account XML document. The document contains some account properties. For example, billName and emailAddress.

**Note:** Smile uses the `key` attribute on the `<Timezone>`, `<AccountTerms>` and `<Costcentre>` elements as an internal identifier.

```
<AccountInfo xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>129512142</USN>
  <Invoicee>
    <InvoiceeUsn>2142421482</InvoiceeUsn>
  </Invoicee>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
```

```
    <CycleDay>20</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <AccountTerms key="1">Default terms</AccountTerms>
  <CostCentre key="105">Example cost centre</CostCentre>
  <Properties>
    <Object>
      <String name="billName">Smith</String>
      <String name="billNumber">60</String>
      <String name="billPostCode">3000</String>
      <String name="billState">VIC</String>
      <String name="billSuburb">Melbourne</String>
      <String name="billType">ST</String>
      <String name="emailAddress">demo@example.com</String>
      <String name="name">Example Company Pty Ltd</String>
      <String name="workPhone">+61355501256</String>
    </Object>
  </Properties>
</AccountInfo>
```

### Faults

`AuthenticationException`

This fault is returned when the caller does not have permission to access the account.

`SmileApiException`

This fault is returned when the account does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAccountInfo usn
```

***usn***

**Purpose:** Specifies the USN of the account to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `findAccount()`

```
Account findAccount(SearchCondition searchCondition)
```

This method searches by custom field for an account and returns an XML document with detailed information about the account and its associated subscriptions. This method searches only custom fields that you have defined in Smile; it does not search any default properties such as USN or username.

### Parameters

**searchCondition**

**Format:** SearchCondition XML

**Purpose:** Defines a SearchCondition XML document that specifies the custom field attribute and value to find.

### Example: Find an account by custom field

This example searches for an account that has the value JoeSmith1 in the custom field client_id.

The value in the `<Attribute>` element is the code name of the custom field. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Fields**, then click **View** next to the appropriate custom field. The code name is in the **Code** field.

The value in the `<Value>` element is the value in the custom field. If the value is part of a custom collection, specify the code name for the value. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Collections**, then click **View** next to the appropriate custom collection. The code names are listed next to each value in the custom collection.

```
<SearchCondition xmlns="http://xml.inomial.com/smile/2.xsd">
  <Equals>
    <Attribute>client_id</Attribute>
    <Value>JoeSmith1</Value>
  </Equals>
</SearchCondition>
```

### Results

This method returns an Account XML document that contains detailed information about the account and its associated subscriptions. This method returns null if no accounts match the search criteria.

There are no messages defined by this method.

### Example: A returned Account XML document

This example shows an Account XML document. The document contains some account properties (for example, billName, emailAddress and client_id) and the subscriptions linked to the account.

**Note:** Smile uses the `key` attribute on the `<Timezone>`, `<AccountTerms>` and `<Costcentre>` elements as an internal identifier.

```
<Account xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>129512142</USN>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <AccountTerms key="1">Default terms</AccountTerms>
  <CostCentre key="105">Example cost centre</CostCentre>
  <Properties>
    <Object>
      <String name="billName">Smith</String>
      <String name="billNumber">60</String>
      <String name="billPostCode">3000</String>
      <String name="billState">VIC</String>
```

```
      <String name="billSuburb">Melbourne</String>
      <String name="billType">ST</String>
      <String name="emailAddress">demo@example.com</String>
      <String name="name">Example Company Pty Ltd</String>
      <String name="workPhone">+61355501256</String>
      <String name="client_id">JoeSmith1</String>
    </Object>
  </Properties>
  <Subscriptions>
    <Subscription>
      <USN>1249871232</USN>
      <SID>17</SID>
      <ServiceName>ADSL</ServiceName>
      <Parent>129512142</Parent>
      <Properties>
        <Object>
          <String name="username">+61355501256</String>
        </Object>
      </Properties>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <SuspendStatus>Active</SuspendStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <InvoicingCycle>
        <CycleType>Anniversary</CycleType>
        <CycleDay>20</CycleDay>
      </InvoicingCycle>
      <RatingCycle>
        <CycleType>Anniversary</CycleType>
        <CycleDay>20</CycleDay>
      </RatingCycle>
      <Timezone key="19">Australia/Sydney</Timezone>
    </Subscription>
  </Subscriptions>
</Account>
```

### Faults

`TooManyResultsException`

This fault is returned when more than one account matches the search criteria.

`NoSuchItemException`

This fault is returned when the custom field that you are searching for does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindAccount propName value
```

***propName***

**Purpose:** Specifies the custom field code name.

***value***

**Purpose:** Specifies the custom field value to find.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `findAccounts()`

```
AccountSearchResultList findAccounts(SearchCondition searchCondition,
 int limit, int offset)
```

This method searches by custom field for accounts and returns an XML document with detailed information about any matches. This method searches only custom fields that you have defined in Smile; it does not search any default properties such as USN or username.

**Note:** Subscriptions belonging the returned accounts are not displayed.

#### Parameters

**searchCondition**

**Format:** SearchCondition XML

**Purpose:** Defines a SearchCondition XML document that specifies the custom field attribute and value to find.

**limit**

**Format:** integer

**Purpose:** Specifies the maximum number of results to return. Should be a positive integer not exceeding 200.

**offset**

**Format:** integer

**Purpose:** Specifies a zero-based index position from within the result set of the search from where to begin returning results.

**Note:** When calling `findAccounts()` for the first time with a new search, specify zero for this argument.

#### Example: Find accounts by custom field

This example searches for an account that has the value `JoeSmith1` in the custom field `client_id`.

The value in the `<Attribute>` element is the code name of the custom field. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Fields**, then click **View** next to the appropriate custom field. The code name is in the **Code** field.

The value in the `<Value>` element is the value in the custom field. If the value is part of a custom collection, specify the code name for the value. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Collections**, then click **View** next to the appropriate custom collection. The code names are listed next to each value in the custom collection.

```
<SearchCondition xmlns="http://xml.inomial.com/smile/2.xsd">
  <Equals>
    <Attribute>contactName</Attribute>
    <Value>JoeSmith1</Value>
  </Equals>
```

```
</SearchCondition>
```

### Results

This method returns an XML document that contains a list of zero or more elements that matched the search criteria. Each element contains detailed information about that account's properties and status.

This method returns an empty element if no accounts match the search criteria. If there are more matching accounts than are set by the "limit" argument, then the list will be truncated to the specified limit, and the MoreResults attribute of the element will be set to "true". In this case, the additional matching accounts can be retrieved by making a subsequent follow-up call to `findAccounts()` with the "offset" argument advanced by "limit". For example:

```
findAccounts(searchCondition, 200, 0);
findAccounts(searchCondition, 200, 200);
findAccounts(searchCondition, 200, 400);
// ...
```

Until the MoreResults flag becomes "false".

If the offset parameter points beyond the total number of matching accounts for a given search criterion, then an empty element will be returned.

The list of returned accounts will be ordered in a stable manner to allow accurate pagination of results among successive calls to this method, however Inomial reserves the right to alter the ordering algorithm between future Smile version releases without notice.

There are no messages defined by this method.

### Example: A returned AccountSearchResultList XML document

This example shows a AccountSearchResultList XML document. The document contains elements which feature account properties, including the attribute and value searched for.

```
<AccountSearchResultList moreResults="false" limit="100" offset="0">
  <Accounts xsi:type="AccountType" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
    <USN>1000000008</USN>
    <AccountType key="1">Standard Account</AccountType>
    <InvoicingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>31</CycleDay>
    </InvoicingCycle>
    <RatingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>31</CycleDay>
    </RatingCycle>
    <Timezone key="13">Australia/Melbourne</Timezone>
    <AccountTerms key="1">Default terms</AccountTerms>
    <Properties>
      <Object objectId="848" formId="5" displayName="Account service
 extended data">
        <String name="abn">123456</String>
        <String name="billAddress">123 Fake Street</String>
```

```
        <String name="billBuildingName"/>
        <String name="billCountry"/>
        <String name="billFloorNumber"/>
        <String name="billLotNumber"/>
        <String name="billName"/>
        <String name="billNumber">54321</String>
        <String name="billPostCode"/>
        <String name="billPostalDeliveryType"/>
        <String name="billState"/>
        <String name="billSubUnit"/>
        <String name="billSuburb"/>
        <String name="billType"/>
        <String name="billTypeSuffix"/>
        <String name="comments"/>
        <String name="companyPosition"/>
        <String name="contactFamily">Administrator</String>
        <String name="contactGiven">Administrator</String>
        <String name="contactTitle"/>
        <String name="emailAddress"/>
        <String name="fax"/>
        <String name="homePhone">54321</String>
        <String name="jabberAddress"/>
        <String name="mobilePhone">54321</String>
        <String name="name">Administrator, Administrator</String>
        <String name="secretAnswer"/>
        <String name="secretQuestion"/>
        <String name="sex"/>
        <String name="streetAddress"/>
        <String name="streetBuildingName"/>
        <String name="streetCountry"/>
        <String name="streetFloorNumber"/>
        <String name="streetLotNumber"/>
        <String name="streetName"/>
        <String name="streetNumber"/>
        <String name="streetPostalDeliveryType"/>
        <String name="streetPostcode"/>
        <String name="streetState"/>
        <String name="streetSubUnit"/>
        <String name="streetSuburb"/>
        <String name="streetType"/>
        <String name="streetTypeSuffix"/>
        <String name="tradingName"/>
        <String name="workPhone">54321</String>
        <String name="username">1000000008</String>
        <String name="alternateAccountNumber"/>
        <String name="contactName">JoeSmith1</String>
        <Integer name="preferredHomeContactTime"/>
        <Integer name="preferredMobileContactTime"/>
        <Integer name="preferredWorkContactTime"/>
        <Boolean name="billPrintName">true</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
        <Timestamp name="dob"/>
      </Object>
    </Properties>
  </Accounts>
</AccountSearchResultList>
```

### Faults

`InvalidRequestException`

This fault is returned when either:

- the limit is non-positive or greater than 200
- the offset is negative

`NoSuchItemException`

This fault is returned when the custom field that you are searching for does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.indAccounts filename [limit [offset]]
```

***filename***

**Purpose:** Defines the filename of the SearchCondition XML document.

***limit***

**Purpose:** Specifies the maximum number of results to return; can be between 1 and 200. If omitted, then a default of 100 will be assumed.

***offset***

**Purpose:** Specifies a zero-based position index into the search result set from where to start returning results. If omitted, then a default of zero will be assumed.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAccountSummary()`

```
AccountSummary getAccountSummary(string usn)
```

This method searches by USN for an account and returns an XML document with the account balance, treatment level, aged balances and the number of active subscriptions on this account.

The account's treatment level or disposition is the debtor status of the account, for example, settled, overdue or delinquent. By default the aged balances specify the balances outstanding for 0, 30, 60 and 90+ days. However, your company configuration for aging invoices determines the age groups that appear in the document.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which to retrieve the account summary.

### Results

This method returns an AccountSummary XML document with information about the account balance, treatment level and aged balances.

There are no messages defined by this method.

### Example: A returned AccountSummary XML document

This example shows an AccountSummary XML document. The document contains the account balance ($94), treatment level (settled), the aged balance for 0, 30, 60 and 90+ days and 2 active subscriptions. In this example, the account has a 30 day aged balance of $5 credit and a 90+ day aged balance of $99.

```
<AccountSummary xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142421391</USN>
  <AccountBalance formatted="94.00" currency="AUD">$94.00</
AccountBalance>
  <TreatmentLevel>Settled</TreatmentLevel>
  <AgedBalance>
    <AgedBalanceItem>
      <AgeGroup>0</AgeGroup>
      <AgeGroupName>Current</AgeGroupName>
      <Balance formatted="0" currency="AUD">$0.00</Balance>
    </AgedBalanceItem>
    <AgedBalanceItem>
      <AgeGroup>1</AgeGroup>
      <AgeGroupName>30 days</AgeGroupName>
      <Balance formatted="-5.00" currency="AUD">$(5.00)</Balance>
    </AgedBalanceItem>
    <AgedBalanceItem>
      <AgeGroup>2</AgeGroup>
      <AgeGroupName>60 days</AgeGroupName>
      <Balance formatted="0.00" currency="AUD">$0.00</Balance>
    </AgedBalanceItem>
    <AgedBalanceItem>
      <AgeGroup>3</AgeGroup>
      <AgeGroupName>90+ days</AgeGroupName>
      <Balance formatted="99.00" currency="AUD">$99.00</Balance>
    </AgedBalanceItem>
  </AgedBalance>
  <ActiveSubscriptionCount>2</ActiveSubscriptionCount>
</AccountSummary>
```

### Faults

`AccountException`

This fault is returned when the account does not exist or could not be retrieved.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAccountSummary usn
```

***usn***

**Purpose:** Specifies the USN of the account for which to retrieve the account summary.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage accounts

You manage accounts in Smile. However, you can use the API to update accounts. The `update()` method updates an account.

### `update()`

```
Account update(string usn, Account account)
```

This method updates the contact details of an account. For example, an individual or company's service and billing addresses and contact information such as phone, mobile and email address, date of birth and sex. The method also updates data for subscriptions associated with the account.

This method updates elements nested within the <Object> element in an Account XML document, and may also update accessory data. You cannot use this method to update any elements that are outside the <Object> element.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to update.

**account**

**Format:** Account XML

**Purpose:** Defines an Account XML document that specifies the account data to update.

#### Example: Update account properties

This example changes the email address for an account and also updates secondary contact, which is a compound custom field. Note that the Account XML document contains only the property to update.

```
<Account xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>129512142</USN>
  <Properties>
    <Object>
      <String name="emailAddress">demo@example.com</String>
        <Object name="secondaryContact">
          <String name="secondaryContactName">John Smith</String>
          <String name="secondaryContactNo">+61395505665</String>
        </Object>
    </Object>
  </Properties>
</Account>
```

#### Results

This method returns an Account XML document that contains the updated state of the account.

There are no messages defined by this message.

### Example: A returned Account XML document

This example shows an Account XML document returned from the previous example. The document contains account information (for example, billName and emailAddress) and information about the subscriptions linked to the account. Note that the document does not identify which property has changed.

**Note:** Smile uses the `key` attribute on the `<Timezone>`, `<AccountTerms>` and `<Costcentre>` elements as an internal identifier.

```
<Account xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>129512142</USN>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>20</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <AccountTerms key="1">Default terms</AccountTerms>
  <CostCentre key="105">Example cost centre</CostCentre>
  <Properties>
    <Object>
      <String name="billName">Smith</String>
      <String name="billNumber">60</String>
      <String name="billPostCode">3000</String>
      <String name="billState">VIC</String>
      <String name="billSuburb">Melbourne</String>
      <String name="billType">ST</String>
      <String name="emailAddress">demo@example.com</String>
      <String name="name">Example Company Pty Ltd</String>
      <String name="workPhone">+61355501256</String>
      <Object name="secondaryContact">
        <String name="secondaryContactName">John Smith</String>
        <String name="secondaryContactNo">+61395505665</String>
      </Object>
    </Object>
  </Properties>
  <Subscriptions>
    <Subscription>
      <USN>1249871232</USN>
      <SID>17</SID>
      <ServiceName>ADSL</ServiceName>
      <Parent>129512142</Parent>
      <Properties>
        <Object>
          <String name="username">+61355501256</String>
        </Object>
      </Properties>
      <ProvisionStatus>Provisioned</ProvisionStatus>
      <SuspendStatus>Active</SuspendStatus>
      <ActivateStatus>Activated</ActivateStatus>
      <EnableStatus>Enabled</EnableStatus>
      <InvoicingCycle>
        <CycleType>Anniversary</CycleType>
```

```
        <CycleDay>20</CycleDay>
      </InvoicingCycle>
      <RatingCycle>
        <CycleType>Anniversary</CycleType>
        <CycleDay>20</CycleDay>
      </RatingCycle>
      <Timezone key="19">Australia/Sydney</Timezone>
    </Subscription>
  </Subscriptions>
</Account>
```

### Faults

`AccountException`

This fault is returned when the account does not exist or cannot be updated or the caller does not have permission to update the account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateAccount usn filename
```

***usn***

**Purpose:** Specifies the USN of the account to update.

***filename***

**Purpose:** Defines the filename of the Account XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Retrieve transaction information

A transaction is a financial event that affects an account. Retrieving transaction information lets you view detailed information about when the transaction was raised, the transaction value and any items in the transaction.

The following methods retrieve transaction information:

- `getTransaction()` retrieves detailed transaction information.
- `getTransactionDocument()` retrieves rendered versions of transaction information.
- `getTransactionList()` retrieves a summary list of transaction items.

**Note:** For more information about transactions in Smile, see the Listening for financial transactions chapter.

### `getTransaction()`

```
TransactionDetail getTransaction(string transactionNumber)
```

This method searches by transaction number for a transaction and returns an XML document with detailed transaction information. If the transaction is an itemised transaction such as an invoice or itemised credit, the document also contains information about each transaction item.

#### Parameters

**transactionNumber**

**Format:** string

**Purpose:** Specifies the transaction number of the transaction to retrieve.

#### Results

This method returns a TransactionDetail XML document that contains information about the transaction, for example, currency, amount, date and sales report category. If the transaction is an itemised transaction, the document also contains information about each item in the transaction.

There are no messages defined by this method.

#### Example: A returned TransactionDetail XML document

This example shows a TransactionDetail XML document for an invoice, which is an itemised transaction. The document contains transaction information (for example, currency, amount, sales report category) and detailed information about each transaction item (for example, item code, quantity, amount and discount).

Note the following points about the structure of the TransactionDetail XML document:

- If the transaction is a credit note, debit note, surcharge or surcharge reversal, the `<salesReportCategory>` element is nested in the `<TransactionDetail>` element.
- If the transaction is itemised, the `<salesReportCategory>` element is nested in the `<transactionItem>` element.

Because this example shows a TransactionDetail XML document for an invoice, the `<salesReportCategory>` element does not appear in the `<TransactionDetail>` element. However, because an invoice is an itemised transaction, the `<salesReportCategory>` element does appear in the `<transactionItem>` element.

```
<TransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd">
  <version>2.0</version>
  <transactionType>Invoice</transactionType>
  <company>1</company>
  <usn>2142421391</usn>
  <transactionNumber>21442058</transactionNumber>
  <currency>AUD</currency>
  <amount>25.00</amount>
  <accountType>1</accountType>
  <gstAmount>2.27</gstAmount>
  <entryTimestamp>2011-11-16T15:37:42.742+11:00</entryTimestamp>
  <formattedAmount currency="AUD" formatted="$25.00">25.00</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$2.27">2.27</
formattedGstAmount>
  <unallocatedAmount>25.00</unallocatedAmount>
  <enteredByUsn>2142421409</enteredByUsn>
  <logTimestamp>2012-01-10T15:37:42.842+11:00</logTimestamp>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$25.00">25.00</formattedUnallocatedAmount>
  <transactionDate>2011-11-16+11:00</transactionDate>
  <dueDate>2011-11-30+11:00</dueDate>
  <openDate>2011-11-16+11:00</openDate>
  <closeDate>2012-01-10+11:00</closeDate>
  <quoteNumber>21442058</quoteNumber>
  <createdByUsn>2142421409</createdByUsn>
  <discount>0.00</discount>
  <transactionSource>Manual</transactionSource>
  <transactionItem>
    <usn>2142421391</usn>
    <lineNumber>0</lineNumber>
    <itemCode>000286</itemCode>
    <chargeFrom>2011-11-16+11:00</chargeFrom>
    <chargeTo>2011-11-16+11:00</chargeTo>
    <description>Prepaid time purchased</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>22.73</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>2.270000</gstAmount>
    <salesReportCategory>invoice</salesReportCategory>
  </transactionItem>
  <transactionItem>
    <usn>2142421391</usn>
    <lineNumber>1</lineNumber>
    <itemCode>gst</itemCode>
    <chargeFrom>2011-11-16+11:00</chargeFrom>
    <chargeTo>2011-11-16+11:00</chargeTo>
```

```
    <description>Includes 10% GST</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>2.27</amount>
    <isGst>true</isGst>
    <chargeGst>false</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>0.000000</gstAmount>
    <salesReportCategory>invoice</salesReportCategory>
  </transactionItem>
</TransactionDetail>
```

### Faults

`AccountTransactionException`

This fault is returned when the specified transaction number does not exist or cannot be retrieved, or the caller does not have permission to retrieve the transaction.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTransaction transactionNumber
```

***transactionNumber***

**Purpose:** Specifies the transaction number of the transaction to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getTransactionDocument()`

```
DocumentReferenceType getTransactionDocument(string
 transactionNumber)
```

This method provides the ability to access rendered versions, such as PDF and XLS, of transactions such as Invoices and Payments.

### Parameters

**transactionNumber**

**Format:** string

**Purpose:** Specifies the number of the document to be rendered, for example, invoice number.

### Results

This method returns a DocumentReferenceType XML document containing a URL from which you can download the requested document. The URL is valid for 24 hours after it is generated by Smile.

The document will be rendered using the default stationery.

This method and the URL both use basic authentication. You must use the same credentials to access the URL and you did to call the method.

There are no messages defined by this method.

### Example: A returned DocumentRefereneType XML document

This example shows a DocumentReferenceType XML document with URL to the rendered document file.

```
<DocumentReference xmlns="http://xml.inomial.com/smile/2.xsd">
  <DocumentURL>http://test.example.com/api-downloads/
documents/9BA2CFEC-1FF2-4BF6-9CEE-CC55ECFCECDA</DocumentURL>
  <Expiry>2013-05-31T00:00:00.000+10:00</Expiry>
  <ReferenceID>9BA2CFEC-1FF2-4BF6-9CEE-CC55ECFCECDA</ReferenceID>
</DocumentReference>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified transaction number does not exist, cannot be retrieved, or the caller does not have permission to retrieve the transaction.

`NoDefaultStationeryException`

This fault is returned when the specified transaction number does not have a default stationery on the given account.

`AccountTransactionException`

This fault is returned when an unexpected error occurs when fetching the transaction to test access requirements.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTransactionDocument
 transactionNumber
```

***transactionNumber***

**Purpose:** Specifies the transaction number of the transaction to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getTransactionList()`

```
TransactionList getTransactionList(string usn, dateTime
 startTimestamp, dateTime endTimestamp)
```

This method retrieves a summary list of transaction items on the ledger for a given USN.

The returned transaction list does not include itemised line items. Call getTransaction() to return detailed transaction information.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which to retrieve transaction information.

**startTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time that starts the date range, for example, 2012-05-01T00:00:00+11:00.

**endTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time that ends the date range, for example, 2012-05-31T23:59:59+11:00.

### Results

This method returns a TransactionList XML document containing summary information about transactions associated with an account.

There are no messages defined by this method.

### Example: A returned TransactionList XML document

This example shows a TransactionList XML document.

```
<TransactionList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns"http://ws.inomial.com/smile.2">
  <transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="TransactionDetailType">
    <version>2.0</version>
    <transactionType>Receipt</transactionType>
    <company>1</company>
    <usn>2142422787</usn>
    <transactionNumber>21431499</transactionNumber>
    <currency>AUD</currency>
    <amount>-100.00</amount>
    <accountType>1</accountType>
    <gstAmount>0.00</gstAmount>
    <entryTimestamp>2015-01-13T09:37:25.855+11:00</entryTimestamp>
    <formattedAmount currency="AUD" formatted="$(100.00)">-100.00</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
    <unallocatedAmount>-100.00</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2015-01-13T09:37:25.857+11:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$(100.00)">-100.00</formattedUnallocatedAmount>
  </transaction>
  <transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="ItemisedTransactionDetailType">
    <version>2.0</version>
```

```
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>2142422787</usn>
    <transactionNumber>21431481</transactionNumber>
    <currency>AUD</currency>
    <amount>100.00</amount>
    <accountType>1</accountType>
    <gstAmount>9.09</gstAmount>
    <entryTimestamp>2015-01-13T09:40:53.715+11:00</entryTimestamp>
    <formattedAmount currency="AUD" formatted="$100.00">100.00</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$9.09">9.09</
formattedGstAmount>
    <unallocatedAmount>100.00</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2015-01-13T09:40:53.720+11:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$100.00">100.00</formattedUnallocatedAmount>
    <transactionDate>2015-01-13+11:00</transactionDate>
    <dueDate>2015-01-25+11:00</dueDate>
    <openDate>2015-01-13+11:00</openDate>
    <closeDate>2015-01-13+11:00</closeDate>
    <quoteNumber>21431481</quoteNumber>
    <createdByUsn>1000000016</createdByUsn>
    <discount>0.00</discount>
    <transactionSource>Manual</transactionSource>
  </transaction>
</TransactionList>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided account number does not exist.

`InvalidRequestException`

This fault is returned when the endTimestamp is not after the startTimestamp.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTransactionList usn
 startTimestamp endTimestamp
```

***usn***

**Purpose:** Specifies the USN of the account for which to retrieve transaction information.

***startTimestamp***

**Purpose:** Specifies the date and time that starts the date range.

***endTimestamp***

**Purpose:** Specifies the date and time that ends the date range.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Download invoice activity statements

Activity statements, or CDRs, provide an itemised list of subscription usage over a period of time. For example, an activity statement for an account with a mobile subscription contains details of the calls made from the mobile, such as the call destination and duration, and any data charges.

Downloading activity statements helps you integrate Smile into your existing network. For example, you can forward the activity statements to an existing database to integrate Smile with your reporting infrastructure.

The `getRatings()` method downloads rated activity statements for a given invoice.

**Note:** For more information about downloading activity statements for a subscription or account, see the methods on the subscription endpoint.

### `getRatings()`

```
ActivityBatchURL getRatings(string transactionNumber)
```

This method returns a URL from which you can download the rated activity statements for a given invoice. An invoice is a document that contains a list of charges raised against a subscription over a given period of time.

#### Parameters

**transactionNumber**

**Format:** string

**Purpose:** Specifies the transaction number of the invoice whose activity statements you want to download.

#### Results

This method returns a URL from which you can download the rated activity statements. The URL is valid for 24 hours after it is generated by Smile. Activity statements are collated into a CSV file.

If there are no activity statements associated with the invoice, the method returns activity="false".

This method and the URL both use basic access authentication. You must use the same credentials to access the URL as you did to call the method.

There are no messages defined by this method.

**Note:** For more information about the format of activity statements, see the CDR documentation under Format Guides in the Inomial Documentation Library.

### Example: A returned ActivityBatchURL XML document with activity statements

This example shows a returned ActivityBatchURL document with the URL from which you can download the activity statements.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="true">
  <BatchID>1409213</BatchID>
  <URL>https://smile.example.com/activity?token=12049IASUBF9812p9</
URL>
</ActivityBatchURL>
```

### Example: A returned ActivityBatchURL XML document without activity

This example shows a returned ActivityBatchURL document with no activity.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="false"/>
```

### Faults

`AccountException`

This fault is returned when Smile could not produce the activity statement.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetInvoiceRatings
 transactionNumber
```

***transactionNumber***

**Purpose:** Specifies the transaction number of the invoice whose activity statements you want to download.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage account charges

Managing account charges lets you compile charges, raises invoices, move transactions, apply additional charges or adjustments or reverse transactions on an account.

The following methods manage account charge changes:

- `compileChargeList()` compiles a quote of charges for an account.
- `raiseInvoice()` raises an invoice to an account.
- `allocate()` allocates transactions within an account.
- `migrateOpenTransaction()` moves an open transaction to another account.
- `raiseAdjustment()` raises an adjustment against an account.
- `getDeferredCharges()` retrieves a list of deferred charges for a USN.
- `addDeferredCharge()` adds item charges to an account, specifying a future date for when they will be charged.
- `raiseReimbursement()` raises a reimbursement to an account.
- `reverseTransaction()` reverses a transaction.

### `compileChargeList()`

```
NewInvoiceRequest compileChargeList(string usn,
 ChargeListCompilationRequest chargeListCompilationRequest)
```

This method compiles a NewInvoiceRequest from the provided ChargeListCompilationRequest, taking the specified account's context into consideration. For example, for tax or discount purposes.

This method does not affect the account at all. It is provided as a way to estimate charges.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to compile the charge list against.

**chargeListCompilationRequest**

**Format:** ChargeListCompilationRequest XML

**Purpose:** Defines a ChargeListCompilationRequest XML document specifying the details of the charge list.

#### Example: Request a compilation of charges

This example shows a request to compile a charge list for a single charge for internet access and computed GST to be applied to account 2142421532.

```
<ChargeListCompilationRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <ChargeList>
    <Charge>
      <ChargeID>DE8207D0-EB19-496C-978B-5048C096F4A1</ChargeID>
```

```
      <ItemDescriptionID>68</ItemDescriptionID>
      <Description>Internet Access $10/mo</Description>
      <ChargeStart>2014-03-12T00:00:00+11:00</ChargeStart>
      <ChargeEnd>2014-04-12T00:00:00+11:00</ChargeEnd>
      <Count>1</Count>
      <Charge>10.00</Charge>
      <ChargeGST>true</ChargeGST>
      <GSTIsPredetermined>true</GSTIsPredetermined>
      <GSTId>1</GSTId>
      <GSTAmount>1</GSTAmount>
      <USN>2142421532</USN>
    </Charge>
  </ChargeList>
</ChargeListCompilationRequest>
```

### Results

This method returns a NewInvoiceRequest which may be used as a quote for the ChargeList. It can also be provided to `raiseInvoice()`, with possible modifications, to raise against the account.

There are no messages defined by this method.

### Example: A returned NewInvoiceRequest XML document

This example shows a NewInvoiceRequest XML document that contains the compiled internet and GST charges as they would apply to the account, which also receives a 50% discount.

```
<NewInvoiceRequest xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <effectiveDate>2014-12-19+11:00</effectiveDate>
  <dueDate>2014-12-31+11:00</dueDate>
  <ChargeRequest>
    <USN>2142421532</USN>
    <itemCode>hfp</itemCode>
    <description>Internet Access $10/mo</description>
    <count>1</count>
    <startDate>2014-03-12+11:00</startDate>
    <endDate>2014-04-10+10:00</endDate>
    <amount>10.00</amount>
    <chargeGst>true</chargeGst>
    <gst gstId="1">
      <amount>1</amount>
    </gst>
    <chargeId>de8207d0-eb19-496c-978b-5048c096f4a1</chargeId>
    <discount discountID="458">
      <amount>5.00</amount>
    </discount>
  </ChargeRequest>
</NewInvoiceRequest>
```

### Faults

`NoSuchItemException`

This fault is returned when the account does not exist.

`InvalidRequestException`

This fault is returned when the specified USN is an invalid account number.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CompileChargeList usn filename
```

***usn***

**Purpose:** Specifies the USN of the account.

***filename***

**Purpose:** Defines the filename of the ChargeListCompilationRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `raiseInvoice()`

```
ItemisedTransactionDetail raiseInvoice(string usn, NewInvoiceRequest
 newInvoiceRequest)
```

This method raises a invoice against an account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account against which the invoice is raised.

**newInvoiceRequest**

**Format:** NewInvoiceRequest XML

**Purpose:** Defines a NewInvoiceRequest XML document that specifies the details of the invoice.

### Example: A NewInvoiceRequest XML document

This example shows a NewInvoiceRequest XML document.

A NewInvoiceRequest document contains the following elements:

| | |
|---|---|
| **<effectiveDate>** | (Optional) Specifies the invoice's transaction date. |
| **<dueDate>** | (Optional) Specifies the invoice’s due date. |

| | | |
|---|---|---|
| **<ChargeRequest>** | Specifies the details of the charge to include on the invoice. Multiple ChargeRequest elements can be specified for an invoice. | |
| | **<USN>** | Specifies the USN of the account that the charge is to be made against. |
| | **<itemCode>** | Specifies the item code of the item,(or category of item, that is to be billed. |
| | **<description>** | (Optional) Specifies a free-form text description of the item. |
| | **<quantity>** | (Optional) Specifies a volume measurement of the items that are being billed in this line item. |
| | **<count>** | (Optional) Specifies the number of instances that this line item is billed. For example, multiple purchases of the same item. If omitted, 1 is the default. |
| | **<startDate>** | Specifies the starting date of the charge period. |
| | **<endDate>** | Specifies the ending date of the charge period. If the charge is for a once-off item, then set endDate the same as startDate. |
| | **<amount>** | (Optional) Specifies the charge amount, excluding tax. |
| | **<chargeGST>** | (Optional) `true` if GST is to be charged for this item; otherwise `false` otherwise. If omitted, false is the default. |
| | **<gst>** | (Optional). Specifies the amount of GST chargeable. |
| | **<chargeId>** | (Optional). Specifies the unique UUID identifier for this charge. This identifier can be used to correlate this charge |

| | |
|---|---|
| | in other systems and can also be used as reference if this charge is to be later retracted.<br><br>**Note:** The client is responsible for ensuring that this UUID identifier is unique and has never been used for a previous charge. |
| **<retractChargeId>** | (Optional) Specifies the chargeId of a previous charge that is to be retracted at the same time that this charge is lodged. |
| **<discount>** | (Optional) Specifies if a discount applies for this line item. |
| **<purchaseOrderNumber>** | (Optional) Specifies a customer-supplied purchase order number associated with this line item. |
| **<overrideCostCentre>** | (Optional) Specifies a cost centre is to be associated with this line item. By default the account's default cost centre, if any, will be used. |
| **<Object>** | (Optional, unless the item description has one or more mandatory custom fields) Specifies custom fields for this invoice line item if any have been defined for the item description referred to by `<itemCode>`. |
| **<location>** | (Optional) Specifies the point-of-sale location for the transaction. The key="..." attribute should contain the Office ID numeric key listed next to the office location's entry under Office Locations in the Smile Configuration menu. |

| | |
|---|---|
| **`<InvoiceOperation>`** | (Optional) By default or when the value is `Close`, the invoice will be closed. When the value is `LeaveOpen`, the invoice will stay open. |
| **`<suppressAutoPayment>`** | `true` or `false`<br>(Optional) If there are bank details associated with the account and no auto payments should be scheduled, suppressAutoPayment should be set to `true`. |

```
<NewInvoiceRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <dueDate>2015-02-14T00:00:00+11:00</dueDate>
  <ChargeRequest>
    <USN>1000000008</USN>
    <itemCode>cst</itemCode>
    <startDate>2015-01-01T00:00:00+11:00</startDate>
    <endDate>2015-01-01T00:00:00+11:00</endDate>
    <amount>50</amount>
    <chargeGst>true</chargeGst>
    <overrideCostCentre key="568">IT department</overrideCostCentre>
  </ChargeRequest>
  <ChargeRequest>
    <USN>1000000008</USN>
    <itemCode>adsl</itemCode>
    <startDate>2015-01-01T00:00:00+11:00</startDate>
    <endDate>2015-02-22T00:00:00+11:00</endDate>
    <chargeGst>false</chargeGst>
    <Object>
      <String name="colour">Chartreuse</String>
    </Object>
    <location key="12"/>
  </ChargeRequest>
  <InvoiceOperation>Close</InvoiceOperation>
  <suppressAutoPayment>false</suppressAutoPayment>
</NewInvoiceRequest>
```

### Results

This method returns an ItemisedTransactionDetails XML document containing details of the invoice.

Smile emits the TransactionClosed message if a closed invoice is produced. A closed invoice is produced when the `<InvoiceOperation>` element value is `Close` or not present.

The value of the `<effectiveCostCentre>` element will either reflect the `<overrideCostCentre>` if one is defined in the NewInvoiceRequest XML document for a `<transactionItem>`, or the default account cost centre is used instead. If the account lacks a default cost centre, then the `<effectiveCostCentre>` element will be omitted for these transaction items entirely.

### Example: A returned ItemisedTransactionDetail XML document

This example shows an ItemisedTransactionDetail XML document.


```
<ItemisedTransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <version>2.0</version>
  <transactionType>Invoice</transactionType>
  <company>1</company>
  <usn>1000000008</usn>
  <transactionNumber>21441027</transactionNumber>
  <currency>AUD</currency>
  <amount>154.00</amount>
  <accountType>1</accountType>
  <gstAmount>5.00</gstAmount>
  <entryTimestamp>2015-01-19T09:54:39.027+11:00</entryTimestamp>
  <salesReportCategory>invoice</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$154.00">154.00</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$5.00">5.00</
formattedGstAmount>
  <unallocatedAmount>154.00</unallocatedAmount>
  <enteredByUsn>2142421276</enteredByUsn>
  <logTimestamp>2015-01-19T09:54:39.033+11:00</logTimestamp>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$154.00">154.00</formattedUnallocatedAmount>
  <transactionDate>2015-01-19+11:00</transactionDate>
  <dueDate>2015-02-14+11:00</dueDate>
  <openDate>2015-01-19+11:00</openDate>
  <closeDate>2015-01-19+11:00</closeDate>
  <quoteNumber>21441027</quoteNumber>
  <createdByUsn>2142421276</createdByUsn>
  <discount>0.00</discount>
  <transactionSource>Manual</transactionSource>
  <transactionItem>
    <usn>1000000008</usn>
    <lineNumber>0</lineNumber>
    <itemCode>cst</itemCode>
    <chargeFrom>2015-01-01+11:00</chargeFrom>
    <chargeTo>2015-01-01+11:00</chargeTo>
    <description>Consultation Fee</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>50.00</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>5.000000</gstAmount>
    <salesReportCategory>invoice</salesReportCategory>
    <effectiveCostCentre key="568">IT department</
effectiveCostCentre>
  </transactionItem>
  <transactionItem>
    <usn>1000000008</usn>
    <lineNumber>1</lineNumber>
```

```
    <itemCode>adsl</itemCode>
    <chargeFrom>2015-01-01+11:00</chargeFrom>
    <chargeTo>2015-02-22+11:00</chargeTo>
    <description>ADSL2 Modem</description>
    <quantity>1.000</quantity>
    <displayRate>$108.90</displayRate>
    <currency>AUD</currency>
    <amount>99.00</amount>
    <isGst>false</isGst>
    <chargeGst>false</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>0.000000</gstAmount>
    <salesReportCategory>hardware</salesReportCategory>
    <Object formId="646" objectId="3435" displayName="ADSL2 Modem
  (Chartreuse)">
      <String name="colour">Chartreuse</String>
    </Object>
    <location key="12">Greenfields store</location>
  </transactionItem>
  <transactionItem>
    <usn>1000000008</usn>
    <lineNumber>2</lineNumber>
    <itemCode>gst</itemCode>
    <chargeFrom>2015-01-19+11:00</chargeFrom>
    <chargeTo>2015-01-19+11:00</chargeTo>
    <description>Includes 10% GST</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>5.00</amount>
    <isGst>true</isGst>
    <chargeGst>false</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>0.000000</gstAmount>
    <salesReportCategory>gst</salesReportCategory>
  </transactionItem>
</ItemisedTransactionDetail>
```


### Faults

`NoSuchItemException`

This fault is returned when the account does not exist or the USN, itemcode , GSTIds, discountIds or retractChargeIds in the ChargeRequest cannot be found.

`InvalidRequestException`

This fault is returned when the USN or new invoice request are missing, the USN in ChargeRequest does not belong to the account, or the request is invalid for another reason.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RaiseInvoice usn filename
```

***usn***

**Purpose:** Specifies the USN of the account against which the invoice is raised.

***filename***

**Purpose:** Defines the filename of the NewInvoiceRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `allocate()`

```
void allocate(string usn, AllocationRequests allocationRequests)
```

This method allocates transactions within an account. A single transaction can be allocated to multiple transactions with different amount values.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to allocate transactions.

**allocationRequests**

**Format:** AllocationRequests XML

**Purpose:** Defines an AllocationRequests XML document that specifies allocation requests.

### Example: Request an allocation

This example shows a request to allocate an amount of $59.90 from transaction 21432521 to transaction 21432513.

An AllocationRequests XML document can have the from transaction allocated to multiple transactions, by including multiple `<AllocateTo>` elements. An AllocationRequests XML document contains the following fields:

| | |
|---|---|
| **`<TransactionNumber>`** | Specifies the transaction number where amounts are allocated from. |
| **`<AutoAllocate>`** | Specifies to use auto-allocation instead of providing `<AllocateTo>` elements. If the transaction numbers where the amount are allocated from are specified, they will be automatically allocated to the most appropriate transaction number. If no transaction numbers are specified, auto-allocation for all debits and credits available for the account is processed. |
| **`<TransactionNumber>`** | Specifies the transaction number where the amount is allocated to. |

**<AllocateAmount>** Specifies the allocation amount.

```
<AllocationRequests xmlns="http://xml.inomial.com/smile/2.xsd">
  <AllocationRequest>
    <TransactionNumber>21432521</TransactionNumber>
    <AllocateTo>
      <TransactionNumber>21432513</TransactionNumber>
      <AllocateAmount>59.90</AllocateAmount>
    </AllocateTo>
  </AllocationRequest>
</AllocationRequests>
```

### Example: Request an auto-allocation

This example shows a request to auto-allocate all unallocated transactions within an account.

```
<AllocationRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <AutoAllocate>true</AutoAllocate>
</AllocationRequest>
```

### Example: Request to auto-allocate a transaction to available credits/debits

This examples shows a request to automatically allocate transaction 21432521 to available credits and debits.

```
<AllocationRequests xmlns="http://xml.inomial.com/smile/2.xsd">
  <AllocationRequest>
    <TransactionNumber>21432521</TransactionNumber>
    <AutoAllocate>true</AutoAllocate>
  </AllocationRequest>
</AllocationRequests>
```

### Example: Request to allocate payments to multiple invoices

This example shows a request to allocate separate amounts from transaction 21432521 to invoices 21432513 ($20), 21432539 ($10), and 21432547 ($29.90).

**Note:** Credit transactions can allocated to debit transactions and debit transactions to credit transactions in a request. The `<AllocateTo>` transaction must be the opposite type of the allocate from transaction.

```
<AllocationRequests xmlns="http://xml.inomial.com/smile/2.xsd">
  <AllocationRequest>
    <TransactionNumber>21432521</TransactionNumber>
    <AllocateTo>
      <TransactionNumber>21432513</TransactionNumber>
      <AllocateAmount>20</AllocateAmount>
    </AllocateTo>
    <AllocateTo>
      <TransactionNumber>21432539</TransactionNumber>
      <AllocateAmount>10</AllocateAmount>
    </AllocateTo>
```

```
    <AllocateTo>
      <TransactionNumber>21432547</TransactionNumber>
      <AllocateAmount>29.90</AllocateAmount>
    </AllocateTo>
  </AllocationRequest>
</AllocationRequests>
```

### Results

This method does not return any output.

Smile emits the `TransactionAllocated` message when the allocate task is complete.

### Faults

`NoSuchItemException`

This fault is returned when:

- the allocate from transaction number does not exist
- the account does not exist
- the allocate to transaction number does not exist

`InvalidRequestException`

This fault is returned when:

- the allocate from transaction number is not provided
- the allocate from transaction number does not belong to the account
- the allocate to transaction number does not belong to the account
- the allocate to transaction number has a pending payment
- the allocate amount exceeds the unallocated amount of the allocate to transaction
- the allocate amount exceeds the unallocated amount of the allocate from transaction
- allocating a non positive amount
- when attempting to allocate an invoice reversal to a non-invoice
- attempting to allocate a receipt reversal to a non-receipt
- attempting to allocate a surcharge reversal to a non-surcharge
- attempting to allocate a bond reversal to a non-bond
- partially reversing an invoice, receipt or surcharge

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.Allocate usn filename
```

***usn***

**Purpose:** Specifies the USN of the account to allocate transactions.

***filename***

**Purpose:** Defines the filename of the AllocationRequests XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `migrateOpenTransaction()`

```
ItemisedTransactionDetail migrateOpenTransaction(string
 openTransactionNumber, string targetAccountUsn)
```

This method migrates the specified open transaction to another account.

Only compatible accounts are allowed to migrate open transactions. To be compatible, they must have the same account type, currency and GST schedule.

This method only supports transaction types of quote, itemised credit and itemised debit.

#### Parameters

**openTransactionNumber**

**Format:** string

**Purpose:** Specifies the number of the open transaction that will be migrated.

**targetAccountUsn**

**Format:** string

**Purpose:** Specifies the account that the open transaction will be migrated to.

#### Results

This method returns an ItemisedtransactionDetail XML document that contains details of the migrated transaction.

There are no messages defined by this method.

#### Example: A returned ItemisedTransactionDetail XML document

This example shows an ItemisedTransactionDetail XML document.

```
<ItemisedTransactionDetail xmlns="http://xml.inomial.com/
smile/2.xsd">
  <version>2.0</version>
  <transactionType>Quote</transactionType>
  <company>1</company>
  <usn>2142425111</usn>
  <transactionNumber>21446927</transactionNumber>
  <currency>AUD</currency>
  <amount>108.90</amount>
  <accountType>1</accountType>
  <gstAmount>9.90</gstAmount>
  <salesReportCategory>invoice</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$108.90">108.90</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$9.90">9.90</
formattedGstAmount>
  <unallocatedAmount>108.90</unallocatedAmount>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$108.90">108.90</formattedUnallocatedAmount>
  <transactionDate>2016-07-06+10:00</transactionDate>
  <openDate>2016-07-06+10:00</openDate>
  <quoteNumber>21446927</quoteNumber>
```

```
  <createdByUsn>2142420831</createdByUsn>
  <transactionSource>Manual</transactionSource>
  <transactionItem>
    <usn>2142425129</usn>
    <lineNumber>0</lineNumber>
    <itemCode>adsl</itemCode>
    <chargeFrom>2016-07-06+10:00</chargeFrom>
    <chargeTo>2016-07-06+10:00</chargeTo>
    <description>ADSL2 Modem</description>
    <quantity>1</quantity>
    <currency>AUD</currency>
    <amount>99.00</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0</discount>
    <count>0</count>
    <gstAmount>9.90</gstAmount>
    <salesReportCategory>hardware</salesReportCategory>
    <chargeId>0d8c57c3-5ef3-45c2-bc4e-8e3d541c4283</chargeId>
    <location key="1">Head office</location>
  </transactionItem>
  <transactionItem>
    <usn>2142425111</usn>
    <lineNumber>1</lineNumber>
    <itemCode>gst</itemCode>
    <chargeFrom>2016-07-06+10:00</chargeFrom>
    <chargeTo>2016-07-06+10:00</chargeTo>
    <description>Includes 10% GST</description>
    <quantity>1</quantity>
    <currency>AUD</currency>
    <amount>9.90</amount>
    <isGst>true</isGst>
    <chargeGst>false</chargeGst>
    <discount>0</discount>
    <count>0</count>
    <gstAmount>0.00</gstAmount>
    <salesReportCategory>gst</salesReportCategory>
  </transactionItem>
</ItemisedTransactionDetail>
```

### Faults

`InvalidRequestException`

This fault is returned when:

- the target account USN is not provided
- the target USN is not an account USN
- the open transaction number is not provided
- the transaction with the specified number is not open
- migrating an open transaction of unsupported transaction type
- the target account is incompatible with the account of the open transaction

`NoSuchItemException`

This fault is returned when:

- the open transaction number does not exist
- the target account USN does not exist

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.MigrateOpenTransaction
 openTransactionNumber targetAccountUsn
```

***openTransactionNumber***

**Purpose:** Specifies the number of the open transaction that will be migrated.

***targetAccountUsn***

**Purpose:** Specifies the account that the open transaction will be migrated to.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `raiseAdjustment()`

```
TransactionDetail raiseAdjustment(string usn, int adjustmentType,
 dateTime effective, decimal amount, decimal gst, string comment,
 AllocationRequest allocationRequest)
```

This method raises a credit or debit note against a given account. A credit note decreases the account balance, that is, decreases the amount a debtor owes; a debit note increases the account balance, that is, increases the amount a debtor owes. You can use adjustments to raise balancing transactions that correct mistakes or compensate customers.

The value of the credit or debit note is added to the account balance, and the adjustment transaction is entered in the account history and appears on invoices, statements and sales reports.

Before you use this method, define the limits of the adjustments that can be raised in Smile.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to raise the adjustment against.

**adjustmentType**

**Format:** integer

**Purpose:** Specifies the adjustment type. You predefine the adjustment type in Smile when you define the adjustment limits. This parameter may affect sales report categories and helps you track and report on the number of adjustments that are raised.

**effective**

**Format:** dateTime

**Purpose:** Specifies the date and time to raise the adjustment, for example, 2011-05-20T14:55:00+11:00. This date must be later than the debtor close date.

**amount**

**Format:** decimal (two decimal places)

**Purpose:** Specifies the value, in dollars and cents and including GST, of the adjusting transaction. The adjustment amount cannot be zero. If you use the nil attribute to make this field null, Smile uses the amount predefined in the adjustment type.

**Note:** A negative amount raises a credit note (decreases an account balance), a positive amount raises a debit note (increases the account balance).

**gst**

**Format:** decimal (two decimal places)

**Purpose:** Specifies the GST to adjust. This parameter must have the same sign as amount. If the amount is nil, Smile uses the amount predefined in the adjustment type.

**comment**

**Format:** string

**Purpose:** (Optional) Explains the reason for the adjustment. Smile stores the comment with the transaction record.

**allocationRequest**

**Format:** AllocationRequest XML

**Purpose:** (Optional) Defines an AllocationRequest XML document that contains the transaction number to raise the adjustment against, such as an invoice or a receipt, and the amount to adjust.

You can also use auto-allocation instead of providing transaction number(s).

### Example: Raise a debit note against an undercharged transaction

This example raises a debit note of $170.00 to correct transaction number 21447701. The debit note increases the account balance of the corresponding account, that is, increases the amount the debtor owes.

```
<AllocationRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <AllocateTo>
    <TransactionNumber>21447701</TransactionNumber>
    <AllocateAmount>170.00</AllocateAmount>
  </AllocateTo>
</AllocationRequest>
```

### Example: Request an auto-allocated payment

This example will cause Smile to automatically allocate the adjustment to any unallocated credits or debits on the account.

```
<AllocationRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <AutoAllocate>true</AutoAllocate>
</AllocationRequest>
```

### Results

This method returns a TransactionDetail XML document that contains information about the adjustment transaction. Smile also emits the `TransactionClosed` message when the adjustment transaction is closed.

### Example: A returned TransactionDetail XML document

This example shows a returned TransactionDetail XML document for a $10 adjustment with $1 GST.

Note the following points about the structure of the TransactionDetail XML document:

- If the transaction is a credit note, debit note, surcharge or surcharge reversal, the `<salesReportCategory>` element is nested in the `<TransactionDetail>` element.
- If the transaction is itemised, the `<salesReportCategory>` element is nested in the `<transactionItem>` element.

Because this example shows a TransactionDetail XML document for a credit note, the `<salesReportCategory>` element appears in the `<TransactionDetail>` element. However, because a credit note is not an itemised transaction, the `<transactionItem>` element does not appear.

```
<TransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd">
  <version>2.0</version>
  <transactionType>CreditNote</transactionType>
  <company>1</company>
  <usn>1240001823</usn>
  <transactionNumber>28139823</transactionNumber>
  <currency>AUD</currency>
  <amount>10.00</amount>
  <accountType>12</accountType>
  <gstAmount>1.00</gstAmount>
  <entryTimestamp>2011-05-20T11:55:00+11:00</entryTimestamp>
  <salesReportCategory>credit</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$10.00">10.00</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$1.00">1.00</
formattedGstAmount>
  <unallocatedAmount>10.00</unallocatedAmount>
  <enteredByUsn>1240000182</enteredByUsn>
  <logTimestamp>2011-05-20T11:57:00+11:00</logTimestamp>
  <glCode>C20122</glCode>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$10.00">10.00</formattedUnallocatedAmount>
</TransactionDetail>
```

### Faults

`AccountTransactionException`

This fault is returned when the caller does not have permission to raise an adjustment against the account, a USN is not provided or the GST is incorrectly entered.

`NoSuchItemException`

This fault is returned when the account does not exist or the adjustment type is not defined.

`IllegalArgumentException`

This fault is returned if the specified USN is not an account, or if a value for a parameter does not conform to the predefined adjustment type.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RaiseAdjustment usn adjustmentType
 effective amount gst [comment] [filename]
```

***usn***

**Purpose:** Specifies the USN of the account to raise the adjustment against.

***adjustmentType***

**Purpose:** Specifies the adjustment type.

***effective***

**Purpose:** Specifies the date and time to raise the adjustment, for example, 2011-05-20T14:55:00+11:00.

***amount***

**Purpose:** Specifies the value, in dollars and cents and including GST, of the adjusting transaction. A negative amount raises a credit note, a positive amount raises a debit note.

***gst***

**Purpose:** Specifies the GST to adjust. This parameter must have the same sign as amount.

***comment***

**Purpose:** (Optional) Explains the reason for the adjustment.

***filename***

**Purpose:** (Optional) Defines the filename of the AllocationRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Related concepts

Transaction types on page 376

## `getDeferredCharges()`

```
DeferredChargeList getDeferredCharges(string usn)
```

This method returns a list of deferred charges for a given USN. Deferred charges are also known as pending charges.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account or subscription for which to return deferred charges.

### Results

This method returns a list of charges for the given USN. If there are no charges a document with no charges will be returned.

If an account USN is given all deferred charges for the account and it's subscriptions will be returned, not just for the account itself.

There are no messages defined by this method.

### Example: A returned DeferredChargeList XML document with charges

This example shows a DeferredChargeList XML document with deferred charges for the given USN.

```
<DeferredChargeList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <TransactionItem>
    <usn>2142423413</usn>
    <lineNumber>0</lineNumber>
    <itemCode>000109</itemCode>
    <chargeFrom>2014-12-15+11:00</chargeFrom>
    <chargeTo>2014-12-15+11:00</chargeTo>
    <description>13XX Flagfall</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>1.00</amount>
    <isGst>false</isGst>
    <chargeGst>true</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>0.100000</gstAmount>
    <salesReportCategory>unknown</salesReportCategory>
  </TransactionItem>
</DeferredChargeList>
```

### Example: A returned DeferredChargeList XML document with no charges

This example shows a DeferredChargeList XML document with no deferred charges for the given USN.

```
<DeferredChargeList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2"/>
```

### Faults

`NoSuchExceptionItem`

This fault is returned when the account or subscription USN does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetDeferredCharges usn
```

***usn***

**Purpose:** Specifies the USN of the account or subscription for which to return deferred charges.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### addDeferredCharge()

```
addDeferredCharge(string usn, ChargeRequest charge)
```

This method adds item charges to an account, specifying a future date for when they will be charged.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account against which the charge should be recorded.

**charge**

**Format:** ChargeRequest XML

**Purpose:** Defines a ChargeRequest XML document that specifies the item charges that will be applied to an account and the date range that they will be applied.

#### Example: Add a deferred charge

This example shows an ADSL item charge that will be applied to an account for the date range 01/01/2013 to 22/02/1013.

```
<ChargeRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142430129</USN>
  <itemCode>adsl</itemCode>
  <description></description>
  <quantity>3</quantity>
  <count>0</count>
  <startDate>2013-01-01T00:00:00+11:00</startDate>
  <endDate>2013-02-22T00:00:00+11:00</endDate>
  <amount>22.22</amount>
  <Object>
    <String name="colour">Chartreuse</String>
  </Object>
  <location key="12"/>
</ChargeRequest>
```

#### Results

This method returns the created TransactionItem.

There are no messages defined by this method.

#### Example: A returned DeferredCharge XML document

This example shows a DeferredCharge XML document.

```
<DeferredCharge xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <usn>1000000008</usn>
  <lineNumber>0</lineNumber>
  <itemCode>adsl</itemCode>
  <chargeFrom>2013-12-31+10:00</chargeFrom>
```

```
  <chargeTo>2014-02-21+10:00</chargeTo>
  <description></description>
  <quantity>3</quantity>
  <currency>AUD</currency>
  <amount>22.22</amount>
  <isGst>false</isGst>
  <chargeGst>true</chargeGst>
  <discount>0</discount>
  <count>0</count>
  <gstAmount>2.22</gstAmount>
  <salesReportCategory>hardware</salesReportCategory>
  <Object formId="646" objectId="3435" displayName="ADSL2 Modem
 (Chartreuse)">
    <String name="colour">Chartreuse</String>
  </Object>
  <location key="12">Greenfields store</location>
</DeferredCharge>
```

### Faults

`NoSuchItemException`

This fault is returned when the item being added does not exist.

`InvalidRequestException`

This fault is returned when you submit a request with inconsistent information or missing required fields.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddDeferredCharge usn filename
```

***usn***

**Purpose:** Specifies the USN of the account to add the charge to.

***filename***

**Purpose:** Defines the filename of the ChargeRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### raiseReimbursement()

```
TransactionDetail raiseReimbursement(string usn, ReimbursementRequest
 reimbursementRequest)
```

This method raises a reimbursement receipt to an account. A reimbursement is an amount refunded to an account or subscription that has already been paid.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which to raise the reimbursement.

**reimbursementRequest**

**Format:** ReimbursementRequest XML

**Purpose:** Defines a ReimbursementRequest XML document that specifies information about the reimbursement to be raised.

#### Example: A ReimbursementRequest XML document

This example shows a ReimbursementRequest XML document containing information about the reimbursement to raise.

```
<ReimbursementRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <Amount>50.0</Amount>
  <Comment>Return of bond</Comment>
  <EffectiveDate>2015-01-01+10:00</EffectiveDate>
</ReimbursementRequest>
```

#### Results

This method returns a TransactionDetail XML document containing details about the reimbursement.

Smile emits the `TransactionClosed` message when the reimbursement transaction is closed.

#### Example: A returned TransactionDetail XML document

This example shows a TransactionDetail XML document.

```
<TransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <version>2.0</version>
  <transactionType>Reimbursement</transactionType>
  <company>1</company>
  <usn>1000000008</usn>
  <transactionNumber>21441035</transactionNumber>
  <currency>AUD</currency>
  <amount>50.0</amount>
  <accountType>1</accountType>
  <gstAmount>0</gstAmount>
  <entryTimestamp>2015-01-01T01:00:00.000+11:00</entryTimestamp>
  <salesReportCategory>reibt</salesReportCategory>
```

```
  <formattedAmount currency="AUD" formatted="$50.00">50.0</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$0.00">0</
formattedGstAmount>
  <unallocatedAmount>50.0</unallocatedAmount>
  <enteredByUsn>2142421276</enteredByUsn>
  <logTimestamp>2015-01-19T10:52:36.911+11:00</logTimestamp>
  <description>Return of bond</description>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$50.00">50.0</formattedUnallocatedAmount>
</TransactionDetail>
```

### Faults

`NoSuchItemException`

This fault is returned when the account does not exist.

`InvalidRequestException`

This fault is returned when the USN is not provided, the amount specified is invalid or the effective date is not provided.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RaiseReimbursement usn filename
```

***usn***

**Purpose:** Specifies the USN of the account for which to raise the reimbursement.

***filename***

**Purpose:** Define the name of the ReimbursementRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### reverseTransaction()

```
TransactionDetail reverseTransaction(string usn,
 ReverseTransactionRequest reverseTransactionRequest)
```

This method raises a new transaction that reverses the a specified transaction.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which the reversal is requested.

**reverseTransactionRequest**

**Format:** ReverseTransactionRequest XML

**Purpose:** Defines a ReverseTransactionRequest XML document that specifies the transaction number to be reversed.

#### Example: A ReverseTransactionRequest XML document

This example shows a ReverseTransactionRequest XML document for transaction number 21431481.

A ReverseTransactionRequest XML document contains the following elements:

| | |
|---|---|
| **<Number>** | Specifies the transaction number to reverse. |
| **<SalesReportCategory>** | Specifies the sales report item code to apply to the reversal transaction. |
| **<ReversalDate>** | Specifies the date of the reversal transaction. |
| **<Comment>** | (Optional) Specifies comments or notes about the reversal. |

```
<ReverseTransactionRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Number>21431481</Number>
  <SalesReportCategory>rinvoice</SalesReportCategory>
  <ReversalDate>2015-01-13+10:00</ReversalDate>
  <Comment>Test Reverse Invoice</Comment>
</ReverseTransactionRequest>
```

#### Results

This method returns a TransactionDetail XML document that contains information about the transaction reversal.

Smile emits a TransactionClosed message for the reversal and a TransactionReversed message for the original transaction.

### Example: A returned TransactionDetail XML document

This example shows a TransactionDetail XML document that contains information about the transaction reversal.

```
<TransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <version>2.0</version>
  <transactionType>InvoiceReversal</transactionType>
  <company>1</company>
  <usn>2142422787</usn>
  <transactionNumber>21431523</transactionNumber>
  <currency>AUD</currency>
  <amount>-100.00</amount>
  <accountType>1</accountType>
  <gstAmount>-9.09</gstAmount>
  <entryTimestamp>2015-01-13T11:47:02.749+11:00</entryTimestamp>
  <salesReportCategory>rinvoice</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$(100.00)">-100.00</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$(9.09)">-9.09</
formattedGstAmount>
  <unallocatedAmount>0.00</unallocatedAmount>
  <enteredByUsn>2142420815</enteredByUsn>
  <logTimestamp>2015-01-13T11:47:02.749+11:00</logTimestamp>
  <referencedNumber>21431481</referencedNumber>
  <description>Test Reverse Invoice</description>
  <formattedUnallocatedAmount currency="AUD" formatted="$0.00">0.00</
formattedUnallocatedAmount>
</TransactionDetail>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided transaction number does not exist.

`InvalidRequestException`

This fault is returned when the account number is not provided.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ReverseTransactionRequest usn filename
```

***usn***

**Purpose:** Specifies the USN of the account for which the reversal is requested.

***filename***

**Purpose:** Defines the filename of the ReverseTransactionRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage cost centres

Managing cost centres lets you create, update and delete cost centre information on an account.

The following methods manage cost centres:

- `createCostCentre()` creates a new cost centre for an account.
- `getCostCentre()` retrieves cost centre information for a given ID.
- `getCostCentres()` retrieves a list of costs centres for an account.
- `findCostCentre()` searches cost centres of an account by cost centre name.
- `updateCostCentre()` updates cost centre information on an account.
- `deleteCostCentre()` deletes an unused cost centre form an account.

### `createCostCentre()`

```
CostCentre createCostCentre(NewCostCentre newCostCentre)
```

This method creates a new cost centre on an account. The cost centre limit of an account is 500.

#### Parameters

**newCostCentre**

**Format:** NewCostCentre XML

**Purpose:** Defines a NewCostCentre XML document specifying a new cost centre for an account.

#### Example: Parameter title

This example shows a NewCostCentre XML document to create a new cost centre on account 2142420930.

```
<NewCostCentre>
  <AccountUSN>2142420930</AccountUSN>
  <Name>Melbourne Office</Name>
</NewCostCentre>
```

#### Results

This method returns a CostCentre XML document containing information about the new cost centre.

There are no messages defined by this method.

#### Example: A returned CostCentre XML document

This example shows a CostCentre XML document containing cost centre information for CostCentreId 794.

```
<CostCentre>
   <CostCentreId>794</CostCentreId>
   <AccountUSN>2142420930</AccountUSN>
   <Name>Melbourne Office</Name>
```

```
</CostCentre>
```

### Faults

`NoSuchItemException`

This fault is returned when given account USN does not exist.

`InvalidRequestException`

This fault is returned when:

- there are 500 existing cost centres in the account
- the name is already in use on the given account

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateCostCentre filename
```

***filename***

**Purpose:** Defines the filename for the NewCostCentre XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getCostCentre()`

```
CostCentre getCostCentre(int costCentreId)
```

This method retrieves cost centre information for a given ID.

### Parameters

**costCentreId**

**Format:** integer

**Purpose:** Specifies the ID of the cost centre.

### Results

This method returns a CostCentre XML document that contains cost centre information for the given ID.

There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when there is no cost centre found with the specified ID.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCostCentre costCentreId
```

***getCostCentre***

**Purpose:** Specifies the ID of the cost centre.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getCostCentres()`

```
CostCentres getCostCentres(string usn)
```

This method returns a list of cost centres that are associated with an account.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of an account for which to retrieve cost centres.

#### Results

This method returns a CostCentres XML document containing a list of cost centres for the given account. If there are no cost centres associated with the account, then the `<CostCentres>` element will be empty.

There are no messages defined by this method.

#### Example: A returned CostCentres XML document

This example shows a CostCentres XML document that contains cost centre information for account 2142420658.

```
<CostCentres xmlns="http://xml.inomial.com/smile/2.xsd">
  <CostCentre>
    <CostCentreId>567</CostCentreId>
    <AccountUSN>2142420658</AccountUSN>
    <Name>Research and Development</Name>
  </CostCentre>
  <CostCentre>
    <CostCentreId>568</CostCentreId>
    <AccountUSN>2142420658</AccountUSN>
    <Name>Marketing</Name>
  </CostCentre>
</CostCentres>
```

#### Faults

`NoSuchItemException`

This fault is returned when the given USN does not correspond to an existing account.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCostCentres usn
```

***usn***

**Purpose:** Specifies the USN of an account for which to retrieve cost centres.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `findCostCentre()`

```
CostCentres findCostCentre(string usn, string costCentreName)
```

This method searches cost centres from an account by name.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the account USN.

**costCentreName**

**Format:** string

**Purpose:** Specifies the cost centre name to search for. Case insensitive.

#### Results

This method returns a CostCentres XML document that contains a list of cost centres that match the name given. If there are no matching cost centres, an empty list will be returned.

There are no messages defined by this method.

#### Example: A returned CostCentres XML document

This example shows a CostCentres XML document containing a list of cost centres that match the given name.

```
<CostCentres>
  <CostCentre>
    <CostCentreId>326</CostCentreId>
    <AccountUSN>2142420658</AccountUSN>
    <Name>Sydney Marketing Office</Name>
  </CostCentre>
</CostCentres>
```

#### Faults

`NoSuchItemException`

This fault is returned when the account with the given USN cannot be found.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindCostCentre usn costCentreName
```

***usn***

**Purpose:** Specifies the account USN.

***costCentreName***

**Purpose:** Specifies the cost centre name to search for.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### updateCostCentre()

```
void updateCostCentre(CostCentre costCentre)
```

This method updates the cost centre given by the CostCentreId field. The Account USN cannot be updated.

#### Parameters

**costCentre**

**Format:** CostCentre XML

**Purpose:** Defines a CostCentre XML document that specifies the updated cost centre information.

#### Example: Update cost centre information

This example shows the cost centre information to update for costCentreId 325.

```
<CostCentre>
  <CostCentreId>325</CostCentreId>
  <AccountUSN>2142420658</AccountUSN>
  <Name>Sydney Marketing</Name>
</CostCentre>
```

#### Results

This method does not return any output for a successful update.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when the cost centre with the given ID does not exist.

`InvalidRequestException`

This fault is returned when:

- the cost centre does not belong to the account USN
- when the name is not provided
- when the name is already in use

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateCostCentre filename
```

***filename***

**Purpose:** Defines the filename of the CostCentre XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `deleteCostCentre()`

```
void deleteCostCentre(int costCentreId)
```

This method deletes an unused cost centre.

#### Parameters

**costCentreId**

**Format:** integer

**Purpose:** Specifies the ID of the cost centre to delete.

#### Results

This method does not return any output for a successful deletion.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when there is no cost centre found with the given ID.

`UnsuccessfulRequestException`

This fault is returned when the given cost centre is in use.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.DeleteCostCentre costCentreId
```

***costCentreId***

**Purpose:** Specifies the ID of the cost centre to delete.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage invoice grouping

Invoice grouping lets you create and modify groups to roll up multiple invoice items on an account to one invoice line item.

The following methods manage invoice grouping:

- `createInvoiceGrouping()` creates a new invoice grouping.
- `deleteInvoiceGrouping()` deletes an invoice grouping.
- `getAvailableInvoiceGroupings()` returns a list of invoice groupings configured to an account.
- `getInvoiceGrouping()` returns the invoice grouping for an ID.
- `updateInvoiceGrouping()` updates an invoice grouping.

The invoice grouping WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/account/invoicegrouping?wsdl`

### Related reference

Invoice grouping endpoint on page 436

### `createInvoiceGrouping()`

```
void createInvoiceGrouping(NewInvoiceGrouping
 newInvoiceGroupingRequest)
```

This method creates a new invoice grouping.

#### Parameters

**newInvoiceGroupingRequest**

**Format:** NewInvoiceGrouping XML

**Purpose:** Defines a NewInvoiceGrouping XML document that specifies the invoice grouping to create.

#### Example: A NewInvoiceGrouping XML document

This example shows a NewInvoiceGrouping XML document.

A NewInvoiceGrouping XML document contains the following optional elements:

| | |
|---|---|
| **`<InvoiceGroupingOverrides>`** | (Optional) Overrides the description on the roll up transaction item and the charge types of which charges will be rolled up. |
| **`<ActiveFrom>`** | (Optional) The date from which charges will be rolled up. |
| **`<ActiveTo>`** | (Optional) The date up to which charges will be rolled up. |

**Note:** If both `<ActiveFrom>` and `<ActiveTo>` fields are blank, charges for any date will be rolled up.

```
<NewInvoiceGrouping xmlns="http://xml.inomial.com/smile/2.xsd">
  <Account>2142424304</Account>
```

```
  <InvoiceGroupingConfiguration key="58ac9ef6-bad2-4584-
b5c2-3eb2639cd5ed">All charges</InvoiceGroupingConfiguration>
  <RollupToSubscription>2142424304</RollupToSubscription>
  <ActiveTo>2015-03-31</ActiveTo>
  <Subscriptions>
    <Subscription>2142424338</Subscription>
    <Subscription>2142424320</Subscription>
  </Subscriptions>
  <InvoiceGroupingOverrides>
    <RollupDescription>Overriding Rollup description {0,date} -
 {1,date}</RollupDescription>
    <ChargeTypes>
      <ChargeType key="6">Manually entered charge</ChargeType>
    </ChargeTypes>
  </InvoiceGroupingOverrides>
</NewInvoiceGrouping>
```

### Results

This method does not return any output.

There are no messages defined by this method.

### Faults

`InvalidRequestException`

This fault is returned when

- the account is not provided
- the invoice grouping configuration reference is not provided
- the ActiveFrom date is not before the ActiveTo date
- the invoice grouping configuration is currently inactive
- the RollupToSubscription is not a subscription of the account
- a Subscription entry is blank
- a listed Subscription is not a subscription of the account
- a listed Subscription has an existing invoice grouping active period overlapping the requested invoice grouping

`NoSuchItemException`

This fault is returned when

- the account does not exist
- the invoice grouping configuration reference does not exist
- a charge type specified in InvoiceGroupingOverrides does not exist
- the RollupToSubscription does not exist
- a listed Subscription does not exist

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateInvoiceGrouping filename
```

***filename***

**Purpose:** Defines the filename of the NewInvoiceGrouping XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getInvoiceGrouping()`

```
InvoiceGroupings getInvoiceGrouping(string invoiceGroupingId)
```

This method returns the invoice grouping for the given ID.

### Parameters

**invoiceGroupingId**

**Format:** string

**Purpose:** Specifies the ID of the invoice grouping to retrieve.

### Results

This method returns an InvoiceGroupings XML document containing information about the invoice grouping.

There are no messages defined by this method.

### Example: A returned InvoiceGroupings XML document

This example shows an InvoiceGroupings XML document.

```
<InvoiceGroupings xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <InvoiceGrouping>
    <InvoiceGroupingId>0e7a9ca0-3e43-4595-bc61-4d78a553e7ba</
InvoiceGroupingId>
    <Account>2142424304</Account>
    <InvoiceGroupingConfiguration>
      <InvoiceGroupingConfigurationId>58ac9ef6-bad2-4584-
b5c2-3eb2639cd5ed</InvoiceGroupingConfigurationId>
      <Name>Roll up all charge types</Name>
      <Company>1</Company>
      <ItemCode>004798</ItemCode>
    </InvoiceGroupingConfiguration>
    <RollupToSubscription>2142424304</RollupToSubscription>
    <Subscriptions>
      <Subscription>2142424312</Subscription>
      <Subscription>2142424320</Subscription>
    </Subscriptions>
    <ActiveTo>2015-03-31+11:00</ActiveTo>
    <InvoiceGroupingOverrides>
      <RollupDescription>Overriding Rollup description {0,date} -
 {1,date}</RollupDescription>
```

```
      <ChargeTypes>
        <ChargeType key="6">Manually entered charge</ChargeType>
      </ChargeTypes>
    </InvoiceGroupingOverrides>
  </InvoiceGrouping>
</InvoiceGroupings>
```

### Faults

`NoSuchItemException`

This fault is returned when an invoice grouping for the invoice grouping ID does not exist or has been deleted.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetInvoiceGrouping
 invoiceGroupingId
```

***invoiceGroupingId***

**Purpose:** Specifies the ID of the invoice grouping.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAvailableInvoiceGroupings()`

```
InvoiceGroupings getAvailableInvoiceGroupings(string usn)
```

This method returns a list of invoice groupings configured to an account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which to return invoice grouping detail.

### Results

This method returns an InvoiceGroupings XML document that contains a list of invoice grouping details.

There are no messages defined by this method.

### Example: A returned InvoiceGroupings XML document

This example shows an InvoiceGroupings XML document.

```
<InvoiceGroupings xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <InvoiceGrouping>
    <InvoiceGroupingId>0e7a9ca0-3e43-4595-bc61-4d78a553e7ba</
InvoiceGroupingId>
    <Account>2142424304</Account>
    <InvoiceGroupingConfiguration>
```

```
      <InvoiceGroupingConfigurationId>58ac9ef6-bad2-4584-
b5c2-3eb2639cd5ed</InvoiceGroupingConfigurationId>
      <Name>Roll up all charge types</Name>
      <Company>1</Company>
      <ItemCode>004798</ItemCode>
    </InvoiceGroupingConfiguration>
    <RollupToSubscription>2142424304</RollupToSubscription>
    <Subscriptions>
      <Subscription>2142424312</Subscription>
      <Subscription>2142424320</Subscription>
    </Subscriptions>
    <ActiveTo>2015-03-31+11:00</ActiveTo>
    <InvoiceGroupingOverrides>
      <RollupDescription>Overriding Rollup description {0,date} -
 {1,date}</RollupDescription>
      <ChargeTypes>
        <ChargeType key="6">Manually entered charge</ChargeType>
      </ChargeTypes>
    </InvoiceGroupingOverrides>
  </InvoiceGrouping>
  <InvoiceGrouping>
    <InvoiceGroupingId>8f8869c6-7496-4550-91c3-b673ab428edc</
InvoiceGroupingId>
    <Account>2142424304</Account>
    <InvoiceGroupingConfiguration>
      <InvoiceGroupingConfigurationId>58ac9ef6-bad2-4584-
b5c2-3eb2639cd5ed</InvoiceGroupingConfigurationId>
      <Name>Roll up all charge types</Name>
      <Company>1</Company>
      <ItemCode>004798</ItemCode>
    </InvoiceGroupingConfiguration>
    <RollupToSubscription>2142424304</RollupToSubscription>
    <Subscriptions>
      <Subscription>2142424338</Subscription>
    </Subscriptions>
  </InvoiceGrouping>
</InvoiceGroupings>
```

### Faults

`NoSuchItemException`

This fault is returned when the account does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAvailableInvoiceGroupings usn
```

***usn***

**Purpose:** Specifies the USN of the account for which to return invoice grouping detail.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### updateInvoiceGrouping()

```
void updateInvoiceGrouping(string invoiceGroupingId,
 NewInvoiceGrouping updateInvoiceGroupingRequest)
```

This method updates an invoice grouping.

#### Parameters

**invoiceGroupingId**

**Format:** string

**Purpose:** Specifies the ID of the invoice grouping to update.

**updateInvoiceGroupingRequest**

**Format:** NewInvoiceGrouping XML

**Purpose:** Defines a NewInvoiceGrouping XML document that specifies the invoice grouping details that will be updated.

#### Example: A NewInvoiceGrouping XML document

This example shows a NewInvoiceGrouping XML document.

A NewInvoiceGrouping XML document contains the following optional elements:

| | |
|---|---|
| **<InvoiceGroupingOverrides>** | (Optional) Overrides the description on the roll up transaction item and the charge types of which charges will be rolled up. |
| **<ActiveFrom>** | (Optional) The date from which charges will be rolled up. |
| **<ActiveTo>** | (Optional) The date up to which charges will be rolled up. |

**Note:** If both `<ActiveFrom>` and `<ActiveTo>` fields are blank, charges for any date will be rolled up.

```
<NewInvoiceGrouping xmlns="http://xml.inomial.com/smile/2.xsd">
  <Account>2142424056</Account>
  <InvoiceGroupingConfiguration key="28b1a75d-b911-4ec3-
a250-6740141ebce8">All charges</InvoiceGroupingConfiguration>
  <RollupToSubscription>2142422878</RollupToSubscription>
  <ActiveFrom>2015-03-24+10:00</ActiveFrom>
  <Subscriptions>
    <Subscription>2142424072</Subscription>
  </Subscriptions>
  <InvoiceGroupingOverrides>
    <RollupDescription>Rollup description {0,date} - {1,date}</
RollupDescription>
    <ChargeTypes>
      <ChargeType key="9">Manually entered charge</ChargeType>
    </ChargeTypes>
  </InvoiceGroupingOverrides>
</NewInvoiceGrouping>
```

### Results

This method does not return any output.

There are no messages defined by this method.

### Faults

`InvalidRequestException`

This fault is returned when

- the account is not provided
- the invoice grouping configuration reference is not provided
- the ActiveFrom date is not before the ActiveTo date
- the invoice grouping configuration is currently inactive
- the RollupToSubscription is not a subscription of the account
- a Subscription entry is blank
- a listed Subscription is not a subscription of the account
- a listed Subscription has an existing invoice grouping active period overlapping the requested invoice grouping

`NoSuchItemException`

This fault is returned when

- the invoice grouping for the ID does not exist
- the account does not exist
- the invoice grouping configuration reference does not exist
- a charge type specified in InvoiceGroupingOverrides does not exist
- the RollupToSubscription does not exist
- a listed Subscription does not exist

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateInvoiceGrouping
 invoiceGroupingId filename
```

***invoiceGroupingId***

**Purpose:** Specifies the ID of the invoice grouping to update.

***filename***

**Purpose:** Defines the filename of the NewInvoiceGrouping XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `deleteInvoiceGrouping()`

```
void deleteInvoiceGrouping(string invoiceGroupingId)
```

This method deletes the invoice grouping so it can not be used anymore.

#### Parameters

**invoiceGroupingId**

**Format:** string

**Purpose:** Specifies the ID of the invoice grouping to delete.

#### Results

This method returns does not return any output.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when an invoice grouping for the invoice grouping ID does not exist or has been deleted.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.DeleteInvoiceGrouping
 invoiceGroupingId
```

**invoiceGroupingId**

**Purpose:** Specifies the ID of the invoice grouping to delete.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage purchase orders

Manging purchase orders lets you create, retrieve and update purchase orders on an account.

The following methods manage purchase orders:

- createPurchaseOrder() returns a purchase order document that has been persisted.
- getPurchaseOrder() returns a purchase order for an account.
- listPurchaseOrders() returns a list of purchase orders for an account.
- updatePurchaseOrder() updates information in a purchase order.

### `createPurchaseOrder()`

```
PurchaseOrder createPurchaseOrder(NewPurchaseOrderRequest
 newPurchaseOrderRequest)
```

This method returns a purchase order document that has been persisted.

#### Parameters

**newPurchaseOrderRequest**

**Format:** NewPurchaseOrderRequest XML

**Purpose:** Defines a NewPurchaseOrderRequest XML document that specifies the details of the purchase order.

#### Example: Request a new purchase order

This example shows a NewPurchaseOrderRequest XML document that specifies the details of the purchase order.

**Note:** The `<AccountUSN>` and `<Number>` elements are required fields.

```
<NewPurchaseOrderRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <AccountUSN>2142426747</AccountUSN>
  <Number>PO123456</Number>
  <ExpiryDate>2015-06-24+10:00</ExpiryDate>
  <Object>
    <String name="purchase_number_note">Note here</String>
  </Object>
  <Subscriptions>
    <SubscriptionUSN>2142426903</SubscriptionUSN>
  </Subscriptions>
  <ExclusiveInvoice>true</ExclusiveInvoice>
</NewPurchaseOrderRequest>
```

#### Results

This method returns a PurchaseOrder XML document.

There are no messages defined by this method.

### Example: A returned PurchaseOrder XML document

This example shows a PurchaseOrder XML document.

```
<PurchaseOrder xmlns="http://xml.inomial.com/smile/2.xsd">
  <AccountUSN>2142426747</AccountUSN>
  <Number>PO123456</Number>
  <ExpiryDate>2015-06-24+10:00</ExpiryDate>
  <Object>
    <String name="purchase_number_note">Note here</String>
  </Object>
  <Subscriptions>
    <SubscriptionUSN>2142426903</SubscriptionUSN>
  </Subscriptions>
  <ExclusiveInvoice>false</ExclusiveInvoice>
  <PurchaseOrderId>b3144086-ad4b-4fbe-b9f7-27dca561e7aa</
PurchaseOrderId>
  <Company>1</Company>
  <Deprecated>false</Deprecated>
</PurchaseOrder>
```

### Faults

`NoSuchItemException`

This fault is returned when:

- the given account USN can not be found
- the given subscription USN can not be found

`InvalidRequestException`

This fault is returned when:

- the USN provided is not an account
- the purchase order number is empty
- the object type values are invalid
- any subscriptions to be added to the purchase order do not belong to the account
- the purchase order number already exists in the account

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreatePurchaseOrder filename
```

***filename***

**Purpose:** Defines the filename of the NewPurchaseOrderRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getPurchaseOrder()

```
PurchaseOrder getPurchaseOrder(string purchaseOrderId)
```

This method returns the account purchase order for the given ID.

#### Parameters

**purchaseOrderId**

**Format:** string

**Purpose:** Specifies the ID of the account purchase order.

#### Results

This method returns a PurchaseOrder XML document containing the account purchase order details.

There are no messages defined by this method.

#### Example: A returned PurchaseOrder XML document

This example shows a PurchaseOrder XML document containing details for purchase order ID b3144086-ad4b-4fbe-b9f7-27dca561e7aa.

```
<PurchaseOrder xmlns="http://xml.inomial.com/smile/2.xsd">
  <AccountUSN>2142426747</AccountUSN>
  <Number>PO123456</Number>
  <ExpiryDate>2015-06-24+10:00</ExpiryDate>
  <Object>
    <String name="purchase_number_note">Note here</String>
  </Object>
  <Subscriptions>
    <SubscriptionUSN>2142426903</SubscriptionUSN>
  </Subscriptions>
  <ExclusiveInvoice>false</ExclusiveInvoice>
  <PurchaseOrderId>b3144086-ad4b-4fbe-b9f7-27dca561e7aa</
PurchaseOrderId>
  <Company>1</Company>
  <Deprecated>false</Deprecated>
</PurchaseOrder>
```

#### Faults

`NoSuchItemException`

This fault is returned when a purchase order with the given ID does not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPurchaseOrder purchaseOrderId
```

***purchaseOrderId***

**Purpose:** Specifies the ID of the account purchase order.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `listPurchaseOrders()`

```
PurchaseOrderList listPurchaseOrders(string account)
```

This method returns a list of purchase orders for a given account, including expired and deprecated purchase orders.

#### Parameters

**account**

**Format:** string

**Purpose:** Specifies the account USN.

#### Results

This method returns a PurchaseOrderList XML document for the given account.

There are no messages defined by this method.

#### Example: A returned PurchaseOrderList XML document

This example shows a PurchaseOrderList XML document containing information about two purchase orders for account 2142426747.

```
<PurchaseOrderList xmlns="http://xml.inomial.com/smile/2.xsd">
  <PurchaseOrder>
    <AccountUSN>2142426747</AccountUSN>
    <Number>PO123456</Number>
    <ExpiryDate>2015-06-24+10:00</ExpiryDate>
    <Object>
      <String name="purchase_number_note">Note here</String>
    </Object>
    <Subscriptions/>
    <ExclusiveInvoice>true</ExclusiveInvoice>
    <PurchaseOrderId>b3144086-ad4b-4fbe-b9f7-27dca561e7aa</
PurchaseOrderId>
    <Deprecated>false</Deprecated>
  </PurchaseOrder>
  <PurchaseOrder>
    <AccountUSN>2142426747</AccountUSN>
    <Number>PO987712</Number>
    <Object/>
    <Subscriptions>
      <SubscriptionUSN>2142428503</SubscriptionUSN>
    </Subscriptions>
    <ExclusiveInvoice>false</ExclusiveInvoice>
    <PurchaseOrderId>2be191ff-9152-451f-8cb5-53c6633464f2</
PurchaseOrderId>
    <Deprecated>false</Deprecated>
  </PurchaseOrder>
</PurchaseOrderList>
```

### Faults

`NoSuchItemException`

This fault is returned when an account with the given USN can not be found.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ListPurchaseOrders account
```

***account***

**Purpose:** Specifies the account USN.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `updatePurchaseOrder()`

```
void updatePurchaseOrder(PurchaseOrder purchaseOrder)
```

This method updates information in a purchase order.

### Parameters

**purchaseOrder**

**Format:** PurchaseOrder XML

**Purpose:** Defines a PurchaseOrder XML document that specifies the details to update.

### Example: Update a purchase order

This example shows a PurchaseOrder XML document containing the details to update in purchase order number PO987712.

```
<PurchaseOrder xmlns="http://xml.inomial.com/smile/2.xsd">
  <AccountUSN>2142426747</AccountUSN>
  <Number>PO987712</Number>
  <Object/>
  <Subscriptions>
    <SubscriptionUSN>2142428503</SubscriptionUSN>
    <SubscriptionUSN>2142428511</SubscriptionUSN>
  </Subscriptions>
  <ExclusiveInvoice>true</ExclusiveInvoice>
  <PurchaseOrderId>2be191ff-9152-451f-8cb5-53c6633464f2</
PurchaseOrderId>
  <Deprecated>false</Deprecated>
</PurchaseOrder>
```

### Results

This method does not return any output. There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when:

- an account with the given AccountUSN can not be found
- a subscription with the given SubscriptionUSN can not be found

`InvalidRequestException`

This fault is returned when:

- the USN provided is not an account
- the purchase order number is empty
- the object type values are invalid
- any subscriptions to be added to the purchase order does not belong to the account
- the purchase order number already exists in the account

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdatePurchaseOrder filename
```

***filename***

**Purpose:** Defines the filename of the PurchaseOrder XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Retrieve future billing estimates

Future billing lets you view potential future charges for an account, without applying them.

When viewing future charges Smile generates a future invoice for a specified rating period, displaying the value of :

- usage charges in the current rating period
- recurring charges and deferred charges for future rating periods

A future invoice assumes that all previous future invoice periods have been invoiced. Only the recurring and deferred charges for a specific invoice period are included in the future invoice for that period.

The `getFutureBilling()` method retrieves a future invoice list for an account, including subscriptions.

### `getFutureBilling()`

```
FutureInvoiceList getFutureBilling(string usn, FutureInvoiceRequest
 request)
```

This method returns a future invoice list for an account, including subscriptions, for given invoicing end date(s).

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account.

**request**

**Format:** FutureInvoiceRequest XML

**Purpose:** Defines a FutureInvoiceRequest XML document that specifies the invoicing end date(s).

#### Example: A FutureInvoiceRequest XML document

This example shows a FutureInvoiceRequest XML document specifying invoicing end dates of 30/06/2015 and 31/07/2015.

```
<FutureInvoiceRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <InvoicingEnd>2015-06-30+10:00</InvoicingEnd>
  <InvoicingEnd>2015-07-31+10:00</InvoicingEnd>
</FutureInvoiceRequest>
```

### Results

This method returns a future invoice list for an account, including subscriptions, for given invoicing end date(s).

There are no messages defined by this method.

### Example: A returned FutureInvoiceList XML document

This example shows a returned FutureInvoiceList XML document for account USN 2142421375. Invoicing end dates 2015-06-30+10.00 and 2015-07-31+10.00 are given as parameters. Subscription USN 2142421417 and 2142421425 belong to this account.


```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
<FutureInvoiceList xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <AccountUSN>2142421375</AccountUSN>
  <FutureInvoice>
    <InvoicingEnd>2015-06-30+10:00</InvoicingEnd>
    <RatingStatus>UpToDate</RatingStatus>
    <Invoice>
      <version>2.0</version>
      <transactionType>Quote</transactionType>
      <company>1</company>
      <usn>2142421375</usn>
      <currency>AUD</currency>
      <amount>72.00</amount>
      <accountType>1</accountType>
      <gstAmount>2.00</gstAmount>
      <salesReportCategory>invoice</salesReportCategory>
      <formattedAmount currency="AUD" formatted="$72.00">72.00</
formattedAmount>
      <formattedGstAmount currency="AUD" formatted="$2.00">2.00</
formattedGstAmount>
      <unallocatedAmount>77.00</unallocatedAmount>
      <formattedUnallocatedAmount currency="AUD"
 formatted="$77.00">77.00</formattedUnallocatedAmount>
      <transactionDate>2015-06-12+10:00</transactionDate>
      <openDate>2015-06-12+10:00</openDate>
      <createdByUsn>2142420773</createdByUsn>
      <transactionSource>Bulk</transactionSource>
      <transactionItem>
        <usn>2142421417</usn>
        <lineNumber>0</lineNumber>
        <itemCode>000285</itemCode>
        <chargeFrom>2015-07-01+10:00</chargeFrom>
        <chargeTo>2015-07-31+10:00</chargeTo>
        <description>Recurring Easy5 (01/07/2015 - 31/07/2015)</
description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>20.00</amount>
        <isGst>false</isGst>
        <chargeGst>true</chargeGst>
        <discount>0</discount>
        <count>0</count>
```

```
        <gstAmount>2.00</gstAmount>
        <salesReportCategory>adsl</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142421417</usn>
        <lineNumber>2</lineNumber>
        <itemCode>gst</itemCode>
        <chargeFrom>2015-06-12+10:00</chargeFrom>
        <chargeTo>2015-06-12+10:00</chargeTo>
        <description>Includes 10% GST</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>2.00</amount>
        <isGst>true</isGst>
        <chargeGst>false</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.00</gstAmount>
        <salesReportCategory>gst</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142421425</usn>
        <lineNumber>1</lineNumber>
        <itemCode>005001</itemCode>
        <chargeFrom>2015-07-01+10:00</chargeFrom>
        <chargeTo>2015-07-31+10:00</chargeTo>
        <description>RTR recurring charge</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>50.00</amount>
        <isGst>false</isGst>
        <chargeGst>false</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.00</gstAmount>
        <salesReportCategory>recurring</salesReportCategory>
      </transactionItem>
    </Invoice>
  </FutureInvoice>
  <FutureInvoice>
    <InvoicingEnd>2015-07-31+10:00</InvoicingEnd>
    <RatingStatus>UpToDate</RatingStatus>
    <Invoice>
      <version>2.0</version>
      <transactionType>Quote</transactionType>
      <company>1</company>
      <usn>2142421375</usn>
      <currency>AUD</currency>
      <amount>50.00</amount>
      <accountType>1</accountType>
      <gstAmount>0.00</gstAmount>
      <salesReportCategory>invoice</salesReportCategory>
      <formattedAmount currency="AUD" formatted="$50.00">50.00</
formattedAmount>
      <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
      <unallocatedAmount>275.00</unallocatedAmount>
```

```
      <formattedUnallocatedAmount currency="AUD"
 formatted="$275.00">275.00</formattedUnallocatedAmount>
      <transactionDate>2015-06-12+10:00</transactionDate>
      <openDate>2015-06-12+10:00</openDate>
      <createdByUsn>2142420773</createdByUsn>
      <transactionSource>Bulk</transactionSource>
      <transactionItem>
        <usn>2142421425</usn>
        <lineNumber>0</lineNumber>
        <itemCode>005001</itemCode>
        <chargeFrom>2015-08-01+10:00</chargeFrom>
        <chargeTo>2015-08-31+10:00</chargeTo>
        <description>RTR recurring charge</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>50.00</amount>
        <isGst>false</isGst>
        <chargeGst>false</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.00</gstAmount>
        <salesReportCategory>recurring</salesReportCategory>
      </transactionItem>
    </Invoice>
  </FutureInvoice>
</FutureInvoiceList>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account.

`InvalidRequestException`

This fault is returned when the specified USN is not an account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetFutureBilling usn filename
```

***usn***

**Purpose:** Specifies the USN of the account.

***filename***

**Purpose:** Defines the filename of the FutureInvoiceRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage bonds

Bonds are a sum that are held as security and will generally be reimbursed at a future date. For example, a bond can be held as a deposit for supplied hardware that will be refunded upon return or as a security deposit to mitigate a potential risk of default when providing credit to a customer. For more information, see the *User Guide*.

Managing bonds lets you apply and release bonds on accounts. You can also retrieve information about bonds available in Smile and applied to an account.

The following methods manage bonds:

- `getBondTypes()` returns information on the bonds available in Smile.
- `getHeldBonds()` returns a list of bonds on an account, including released bonds.
- `createBond()` creates a bond on an account.
- `releaseBond()` releases a bond from an account.

### `getBondTypes()`

```
BondTypes getBondTypes()
```

This method retrieves the bond types that are available in Smile.

#### Parameters

This method does not accept any parameters.

#### Results

This method returns a BondTypes XML document containing information about the bond types available in Smile.

There are no messages defined by this method.

#### Example: A returned BondTypes XML document

This example shows a BondTypes XML document with information about the bond types available in Smile.

```
<BondTypes xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <BondType>
    <BondType>6aecbd7b-cfe4-4b79-9af4-92f5e70564d9</BondType>
    <Company>1</Company>
    <Name>ADSL modem</Name>
    <Description>ADSL modem</Description>
    <Rate formatted="$50.00" currency="AUD">50</Rate>
  </BondType>
  <BondType>
    <BondType>f4eb1d21-e27f-47d5-bd04-9b97509e47e2</BondType>
    <Company>1</Company>
    <Name>Pocket WiFi modem</Name>
    <Description>4G modem</Description>
    <Rate formatted="$150.00" currency="AUD">150</Rate>
```

```
  </BondType>
</BondTypes>
```

### Faults

`NoSuchItemException`

This fault is returned when the SOAP caller does not exist in Smile.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetBondTypes
```

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getHeldBonds()`

```
Bonds getHeldBonds(string usn)
```

This method returns a list of bonds on an account, including released bonds.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account for which to retrieve bond information.

### Results

This method returns a Bonds XML document containing information about bonds on an account.

There are no messages defined by this method.

### Example: A returned Bonds XML document

This example shows a Bonds XML document containing information about bonds on an account.

```
<Bonds xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <ItemisedTransactionDetail>
    <version>2.0</version>
    <transactionType>Bond</transactionType>
    <company>1</company>
    <usn>2142422829</usn>
    <transactionNumber>21431465</transactionNumber>
    <currency>AUD</currency>
    <amount>50.00</amount>
    <accountType>1</accountType>
    <gstAmount>0.00</gstAmount>
    <entryTimestamp>2015-01-12T09:20:01.191+11:00</entryTimestamp>
    <formattedAmount currency="AUD" formatted="$50.00">50.00</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
```

```
    <unallocatedAmount>50.00</unallocatedAmount>
    <enteredByUsn>2142420815</enteredByUsn>
    <logTimestamp>2015-01-13T09:20:01.195+11:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$50.00">50.00</formattedUnallocatedAmount>
    <transactionDate>2015-01-12+11:00</transactionDate>
    <openDate>2015-01-13+11:00</openDate>
    <closeDate>2015-01-13+11:00</closeDate>
    <quoteNumber>21431465</quoteNumber>
    <createdByUsn>2142420815</createdByUsn>
    <discount>0.00</discount>
    <transactionSource>Bond</transactionSource>
    <transactionItem>
      <usn>2142422829</usn>
      <lineNumber>0</lineNumber>
      <itemCode>002103</itemCode>
      <chargeFrom>2015-01-12+11:00</chargeFrom>
      <chargeTo>2015-01-12+11:00</chargeTo>
      <description>ADSL modem</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>50.00</amount>
      <isGst>false</isGst>
      <chargeGst>false</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>bond</salesReportCategory>
    </transactionItem>
  </ItemisedTransactionDetail>
</Bonds>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided USN does not exist.

`InvalidRequestException`

This fault is returned when the account USN is not provided.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetHeldBonds usn
```

***usn***

**Purpose:** Specifies the USN of the account for which to retrieve bond information.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `createBond()`

```
ItemisedTransactionDetail createBond(BondRequest bondRequest)
```

This method creates a bond associated with the given account or subscription.

#### Parameters

**bondRequest**

**Format:** BondRequest XML

**Purpose:** Defines a BondRequest XML document that specifies the bond type to apply to an account or subscription USN.

#### Example: A BondRequest XML document

This example shows a BondRequest XML document.

The value in the `<BondType>` element is a bond type key returned by `getBondTypes()`. The value in the `<Amount>` element will override the amount associated with the bond type.

```
<BondRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142422829</USN>
  <BondType>6aecbd7b-cfe4-4b79-9af4-92f5e70564d9</BondType>
  <EffectiveDate>2015-01-12+10:00</EffectiveDate>
  <Amount>50.00</Amount>
</BondRequest>
```

#### Results

This method returns an ItemisedTransactionDetail XML document that contains information about the bond applied to an account.

Smile emits a TransactionClosed message when the bond is applied.

#### Example: A returned ItemisedTransactionDetail XML document

This example shows an ItemisedTransactionDetail XML document.

```
<ItemisedTransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <version>2.0</version>
  <transactionType>Bond</transactionType>
  <company>1</company>
  <usn>2142422829</usn>
  <transactionNumber>21431465</transactionNumber>
  <currency>AUD</currency>
  <amount>50.00</amount>
  <accountType>1</accountType>
  <gstAmount>0.00</gstAmount>
  <entryTimestamp>2015-01-12T09:20:01.191+11:00</entryTimestamp>
  <salesReportCategory>bond</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$50.00">50.00</
formattedAmount>
```

```
  <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
  <unallocatedAmount>50.00</unallocatedAmount>
  <enteredByUsn>2142420815</enteredByUsn>
  <logTimestamp>2015-01-13T09:20:01.195+11:00</logTimestamp>
  <formattedUnallocatedAmount currency="AUD"
 formatted="$50.00">50.00</formattedUnallocatedAmount>
  <transactionDate>2015-01-12+11:00</transactionDate>
  <openDate>2015-01-13+11:00</openDate>
  <closeDate>2015-01-13+11:00</closeDate>
  <quoteNumber>21431465</quoteNumber>
  <createdByUsn>2142420815</createdByUsn>
  <discount>0.00</discount>
  <transactionSource>Bond</transactionSource>
  <transactionItem>
    <usn>2142422829</usn>
    <lineNumber>0</lineNumber>
    <itemCode>002103</itemCode>
    <chargeFrom>2015-01-12+11:00</chargeFrom>
    <chargeTo>2015-01-12+11:00</chargeTo>
    <description>ADSL modem</description>
    <quantity>1.000</quantity>
    <currency>AUD</currency>
    <amount>50.00</amount>
    <isGst>false</isGst>
    <chargeGst>false</chargeGst>
    <discount>0.00</discount>
    <count>0</count>
    <gstAmount>0.000000</gstAmount>
    <salesReportCategory>bond</salesReportCategory>
  </transactionItem>
</ItemisedTransactionDetail>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN or bond type in the bond request does not exist.

`InvalidRequestException`

This fault is returned when the account USN is not provided in the bond request.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateBond filename
```

***filename***

**Purpose:** Defines the filename of the BondRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `releaseBond()`

```
TransactionDetail releaseBond(string usn, string bondNumber)
```

This method creates a BondRelease transaction to record the release of the bond provided.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to release the bond from.

**bondNumber**

**Format:** string

**Purpose:** Specifies the number of the bond to release.

#### Results

This method returns a TransactionDetail XML document containing information about the bond release.

Smile emits a TransactionClosed message when the bond release is created. A TransactionAllocated message is emitted when the bond release is allocated to the bond. A TransactionAllocated message is emitted with the `<allocationType>` element value of `Deallocation` when previous allocations to the bond are deallocated.

#### Example: A returned TransactionDetail XML document

This example shows a TransactionDetail XML document.

```
<TransactionDetail xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <version>2.0</version>
  <transactionType>BondRelease</transactionType>
  <company>1</company>
  <usn>2142422829</usn>
  <transactionNumber>21431473</transactionNumber>
  <currency>AUD</currency>
  <amount>-50.00</amount>
  <accountType>1</accountType>
  <gstAmount>0.00</gstAmount>
  <entryTimestamp>2015-01-13T09:33:30.081+11:00</entryTimestamp>
  <salesReportCategory>bondrel</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$(50.00)">-50.00</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
  <unallocatedAmount>0.00</unallocatedAmount>
  <enteredByUsn>2142420815</enteredByUsn>
  <logTimestamp>2015-01-13T09:33:30.082+11:00</logTimestamp>
  <referencedNumber>21431465</referencedNumber>
  <formattedUnallocatedAmount currency="AUD" formatted="$0.00">0.00</
formattedUnallocatedAmount>
</TransactionDetail>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided USN or bond number does not exist.

`InvalidRequestException`

This fault is returned when the account USN is not provided, or the USN and bond number do not correspond.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ReleaseBond usn bondNumber
```

***usn***

**Purpose:** Specifies the USN of the account to release the bond from.

***bondNumber***

**Purpose:** Specifies the number of the bond to release.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

# Chapter 5

# Managing subscriptions

## Subscription API overview

A subscription is a single instance of a service that is provided to a customer, for example:

- a mobile number
- a DSL connection
- a CAT (customer access toolkit) subscription

A subscription is linked to an account and is often the source of usage and recurring charges. You can use the subscription endpoint of the SOAP API to:

- retrieve subscription information
- update, cancel, suspend and resume a subscription
- apply a feature to a subscription
- manage plans
- add and remove number and IP range reservations
- download rated and unrated activity statements

Subscription API methods are often linked to messages in the SmileSubscription. The messages help you perform the following tasks:

- order a subscription with an upstream provider
- synchronise subscription changes across your network, for example, provisioning and fulfilment
- control network access
- respond to subscription feature changes

The subscription WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/subscription?wsdl`

The plan change WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/subscription/planchange?wsdl`. The plan change endpoint is only available for BigRating (RTR) subscriptions.

### Related reference

Subscription endpoint on page 460
SmileSubscription on page 489

## Order a subscription

You use methods in the order endpoint of the SOAP API to order a subscription; there are no methods in the subscription endpoint that affect the order process.

Related concepts

Place an order on page 46

## Retrieve subscription information

Retrieving subscription information lets you view the current state of a subscription.

The following methods retrieve subscription information:

- `findUsername()` returns the USN for a given username and service.
- `get()` searches by USN for a subscription and returns information about the subscription's properties and status.
- `getAuthenticationCredentials()` returns the clear password for the specified user.
- `findSubscription()` searches by custom field value for a subscription and returns information about the subscription's properties and status.
- `findSubscriptions()` searches by custom field for non-closed subscriptions and returns information about the sunscription's properties and status.
- `getActiveFeatures()` searches by USN for a subscription and returns information about the subscription's active features.

### `findUsername()`

```
string findUsername(int sid, string username)
```

This method returns the USNs for all non-closed subscriptions associated with a given username and service. After you call this method, you can use the USN and the `get()` method to retrieve detailed information about a subscription. You can call this method at any point after the subscription order is entered.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID. The numerical service ID is located in the **Service (ID)** field of the **Service** tab when you view the service in Smile.

**username**

**Format:** string

**Purpose:** Specifies the username of the account.

#### Results

This method returns a USN for each subscription associated with the given username and service.

There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when Smile cannot find a USN associated with the given username or service.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindUsername sid username
```

***sid***

**Purpose:** Specifies the service ID.

***username***

**Purpose:** Specifies the username of the account.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `get()`

```
Subscription get(string usn)
```

This method searches by USN for a subscription and returns an XML document with detailed information about the subscription's properties and status.

You can call this method at any point after the subscription order is entered.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to retrieve.

### Results

This method returns a Subscription XML document that contains detailed information about the subscription's properties and status.

There are no messages defined by this method.

### Example: A returned Subscription XML document

This example shows a Subscription XML document. The document contains subscription properties (username, given name, family name and number of phone lines) and the subscription's status (provisioned, not suspended, activated and enabled).

**Note:** Smile uses the `key` attribute on the `<Timezone>` and `<Costcentre>` elements as an internal identifier.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>120398123</USN>
  <SID>15</SID>
```

```
  <ServiceName>VOIP</ServiceName>
  <Parent>1204921324</Parent>
  <Properties>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">13</Integer>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <CostCentre key="1">Example cost centre</CostCentre>
</Subscription>
```

### Faults

`GetSubscriptionException`

This fault is returned when the subscription does not exist or the caller does not have permission to access the subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscription usn
```

***usn***

**Purpose:** Specifies the USN of the subscription to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Related concepts

Example: Get a subscription on page 29

### getAuthenticationCredentials()

```
Credentials getAuthenticationCredentials(string usn)
```

This method retrieves a user password.

**Note:** The caller requires "view password" SOAP permission to run this method.

**Note:** Currently only password is supported, but this method is designed to be extensible and additional types may be added in the future.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription of which to retrieve authentication credentials.

#### Results

This method returns a Credentials XML document containing the subscription's password.

There are no messages defined by this method.

#### Example: A returned Credentials XML document

This example shows a Credentials XML document.

```
<Credentials xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <Credential>
    <Type>password</Type>
    <Parameters>
      <Object>
        <String name="password">UKRUqW5n7WR</String>
      </Object>
    </Parameters>
  </Credential>
</Credentials>
```

#### Faults

`NoSuchItemException`

This fault is returned when the account does not exist.

`Access Exception`

This fault is returned when attempting to access the password of a protected account.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAuthenticationCredentials usn
```

***usn***

**Purpose:** Specifies the USN of the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `findSubscription()`

```
Subscription findSubscription(SearchCondition searchCondition)
```

This method searches by custom field for a subscription and returns an XML document with detailed information about the subscription. This method searches only custom fields that you have defined in Smile; it does not search any default properties such as USN or username.

You can call this method at any point in the subscription lifecycle after the subscription order is entered.

#### Parameters

**searchCondition**

**Format:** SearchCondition XML

**Purpose:** Defines a SearchCondition XML document that specifies the custom field attribute and value to find.

#### Example: Find a subscription by custom field

This example searches for a subscription that has the value JoeSmith1 in the custom field client_id.

The value in the `<Attribute>` element is the code name of the custom field. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Fields**, then click **View** next to the appropriate custom field. The code name is in the **Code** field.

The value in the `<Value>` field is the value in the custom field. If the value is part of a custom collection, specify the code name for the value. To find the code name, log in to Smile and click **Configuration and Tools** > **Custom Collections**, then click **View** next to the appropriate custom collection. The code names are listed next to each value in the custom collection.

```
<SearchCondition sinmlns="http://xml.inomial.com/smile/2.xsd">
  <Equals>
    <Attribute>client_id</Attribute>
    <Value>JoeSmith1</Value>
  </Equals>
</SearchCondition>
```

#### Results

This method returns a Subscription XML document that contains detailed information about the subscription's properties and status.

This method returns null if no subscriptions match the search criteria.

There are no messages defined by this method.

### Example: A returned Subscription XML document

This example shows a Subscription XML document. The document contains subscription properties (username, given name, family name, number of phone lines and the custom field client_id) and the subscription's status (provisioned, not suspended, activated and enabled).

**Note:** Smile uses the `key` attribute on the `<Timezone>` and `<Costcentre>` elements as an internal identifier.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>120398123</USN>
  <SID>15</SID>
  <ServiceName>VOIP</ServiceName>
  <Parent>1204921324</Parent>
  <Properties>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">13</Integer>
      <String name="client_id">JoeSmith1</String>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <CostCentre key="1">Example cost centre</CostCentre>
</Subscription>
```

### Faults

`TooManyResultsException`

This fault is returned when more than one subscription matches the search criteria.

`NoSuchItemException`

This fault is returned when the custom field that you are searching for does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindSubscription propName value
```

***propName***

**Purpose:** Specifies the custom field code name.

***value***

**Purpose:** Specifies the custom field value to find.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `findSubscriptions()`

```
SubscriptionSearchResultList findSubscriptions(SearchCondition
 searchCondition, int limit, int offset)
```

This method searches by custom field for a non-closed subscription and returns an XML document with detailed information about the matching subscriptions. This method searches only custom fields that you have defined in Smile; it does not search any default properties such as USN or username.

You can call this method at any point in the subscription lifecycle after the subscription order is entered.

#### Parameters

**searchCondition**

**Format:** SearchCondition XML

**Purpose:** Defines a SearchCondition XML document that specifies the custom field attribute and value to find.

**limit**

**Format:** integer

**Purpose:** Specifies the maximum number of results to return. Should be a positive integer not exceeding 200.

**offset**

**Format:** integer

**Purpose:** Specifies a zero-based index position from within the result set of the search from where to begin returning results.

**Note:** When calling `findSubscriptions()` for the first time with a new search, specify zero for this argument.

#### Example: Find subscriptions by custom field

This example searches for a subscription that has the value JoeSmith1 in the custom field client_id.

The value in the `<Attribute>` element is the code name of the custom field. To find the code name, log in to Smile and click **Configuration and Tools > Custom Fields**, then click **View** next to the appropriate custom field. The code name is in the **Code** field.

The value in the `<Value>` field is the value in the custom field. If the value is part of a custom collection, specify the code name for the value. To find the code name, log in to Smile and click **Configuration and**

**Tools** > **Custom Collections**, then click **View** next to the appropriate custom collection. The code names are listed next to each value in the custom collection.

```
<SearchCondition xmlns="http://xml.inomial.com/smile/2.xsd">
  <Equals>
    <Attribute>client_id</Attribute>
    <Value>JoeSmith1</Value>
  </Equals>
</SearchCondition>
```

### Results

This method returns a SubscriptionSearchResultList XML document that contains a list of zero or more `<Subscription>` elements that matched the search criteria.

Each `<Subscription>` element contains detailed information about that subscription's properties and status.

This method returns an empty `<SubscriptionSearchResultList>` element if no subscriptions match the search criteria.

If there are more matching subscriptions than that set by the "limit" argument, then the list will be truncated to the specified limit, and the MoreResults attribute of the `<SubscriptionSearchResultList>` element will be set to "true".

In this case, the additional matching subscriptions can be retrieved by making a subsequent follow-up call to `findSubscriptions()` with the "offset" argument advanced by "limit". For example:

```
findSubscriptions(searchCondition, 200, 0);
findSubscriptions(searchCondition, 200, 200);
findSubscriptions(searchCondition, 200, 400);
// ...
```

Until the MoreResults flag becomes "false".

If the offset parameter points beyond the total number of matching subscriptions for a given search criterion, then an empty SubscriptionSearchResultList will be returned.

It is guaranteed that the list of returned subscriptions will always be ordered in a stable manner to allow accurate pagination of results among successive calls to this method, however Inomial reserves the right to alter the ordering algorithm between future Smile version releases without notice.

There are no messages defined by this method.

### Example: A returned SubscriptionSearchResultList XML document

This example shows a SubscriptionSearchResultList XML document. The document contains two examples of the Subscription element which features subscription properties (username, given name, family name, number of phone lines and the custom field client_id) and the subscription's status (provisioned, not suspended, activated and enabled).

**Note:** Smile uses the key attribute on the `<Timezone>` and `<Costcentre>` elements as an internal identifier.

```
<SubscriptionSearchResultList xmlns="http://xml.inomial.com/
smile/2.xsd" moreResults="false" limit="200" offset="0">
  <Subscription>
    <USN>120398123</USN>
    <SID>15</SID>
    <ServiceName>VOIP</ServiceName>
    <Parent>1204921324</Parent>
    <Properties>
      <Object>
        <String name="username">+61355501256</String>
        <String name="contactGiven">Joe</String>
        <String name="contactFamily">Smith</String>
        <Integer name="phoneLines">13</Integer>
        <String name="client_id">JoeSmith1</String>
        <String name="voipRouterType">Acme 2000</String>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <InvoicingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>24</CycleDay>
    </InvoicingCycle>
    <RatingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>24</CycleDay>
    </RatingCycle>
    <Timezone key="19">Australia/Sydney</Timezone>
    <CostCentre key="1">Example cost centre</CostCentre>
  </Subscription>
  <Subscription>
    <USN>120398134</USN>
    <SID>15</SID>
    <ServiceName>VOIP</ServiceName>
    <Parent>1204921324</Parent>
    <Properties>
      <Object>
        <String name="username">+61355501357</String>
        <String name="contactGiven">John</String>
        <String name="contactFamily">Doe</String>
        <Integer name="phoneLines">14</Integer>
        <String name="client_id">JohnDoe1</String>
        <String name="voipRouterType">Acme 2000</String>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <InvoicingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>24</CycleDay>
```

```
    </InvoicingCycle>
    <RatingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>24</CycleDay>
    </RatingCycle>
    <Timezone key="19">Australia/Sydney</Timezone>
    <CostCentre key="1">Example cost centre</CostCentre>
  </Subscription>
</SubscriptionSearchResultList>
```

### Faults

`InvalidRequestException`

This fault is returned when either:

- the limit is non-positive or greater than 200
- the offset is negative

`NoSuchItemException`

This fault is returned when the custom field that you are searching for does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindSubscriptions propName value
 [limit [offset]]
```

***propName***

**Purpose:** Specifies the custom field code name.

***value***

**Purpose:** Specifies the custom field value to find.

***limit***

**Purpose:** Specifies the maximum number of results to return; can be between 1 and 200. If omitted, then a default of 100 will be assumed.

***offset***

**Purpose:** Specifies a zero-based position index into the search result set from where to start returning results. If omitted, then a default of zero will be assumed.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getConnectionHistory()

```
ConnectionList getConnectionHistory(string usn,
 ConnectionHistoryQuery connectionHistoryQuery, int limit, int
 offset)
```

This method returns a connection list that contains the history of a subscription, for a given date range. The result returned can be limited with limit and offset.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription or account.

**connectionHistoryQuery**

**Format:** ConnectionHistoryQuery XML

**Purpose:** Defines a ConnectionHistoryQuery XML document that specifies the date range for the connection history.

**limit**

**Format:** integer

**Purpose:** Specifies the number of returned results. Default is 200.

**offset**

**Format:** integer

**Purpose:** (Optional) Specifies the starting point of results returned. Default is 0.

#### Example: A ConnectionHistoryQuery XML document

This example shows a ConnectionHistoryQuery XML document displaying the start and end dates of the connection query.

```
<ConnectionHistoryQuery xmlns="http://xml.inomial.com/smile/2.xsd">
  <Start>2013-01-01T00:00:00+11:00</Start>
  <end>2013-04-01T00:00:00+11:00</end>
</ConnectionHistoryQuery>
```

#### Results

This method returns a ConnectionList XML document that contains information about the connection history of a subscription for a given date range.

If additional results exist for the subscription within the date range, the method returns moreResultExist="true".

There are no messages defined by this method.

### Example: A returned ConnectionList XML document with two connection historys and has no more existing results

This example shows a returned ConnectionList XML document with two connection historys.

```
<ConnectionList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2" moreResultExist="false">
  <Connection>
    <Start>2013-01-11T14:11:30.647+11:00<Start>
    <End>2013-01-11T15:01:30.647+11:00<End>
    <DisconnectReason>Unknown<DisconnectReason>
    <Bytesin>1000<Bytesin>
    <Bytesout>1000<Bytesout>
    <Seconds>3000<Seconds>
  <Connection>
  <Connection>
    <Start>2013-04-18T09:00:30.000+10:00<Start>
    <End>2013-04-18T14:33:50.000+10:00<End>
    <DisconnectReason>Unknown<DisconnectReason>
    <Bytesin>3000<Bytesin>
    <Bytesout>10000<Bytesout>
    <Seconds>20000<Seconds>
  <Connection>
<ConnectionList>
```

### Example: A returned ConnectionList XML document with connection history, and has no more existing results

This example shows a returned ConnectionList XML document with limit and offset set.

```
<ConnectionList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2" moreResultExist="true">
  <Connection>
    <Start>2013-01-11T14:11:30.647+11:00<Start>
    <End>2013-01-11T15:01:30.647+11:00<End>
    <DisconnectReason>Unknown<DisconnectReason>
    <Bytesin>1000<Bytesin>
    <Bytesout>1000<Bytesout>
    <Seconds>3000<Seconds>
  <Connection>
<ConnectionList>
```

### Example: A returned ConnectionList XML document without connection history

This example shows a returned ConnectionList document with no history.

```
<ConnectionList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2" moreResultExist="false"/>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetConnectionHistory usn
 connectionHistoryQuery limit [offset]
```

***usn***

**Purpose:** Specifies the USN of the subscription or the account.

***connectionHistoryQuery***

**Purpose:** Defines the filename of the ConnectionHistoryQuery XML document.

***limit***

**Purpose:** Specifies the number of returned results.

***offset***

**Purpose:** (Optional) Specifies the starting point of results returned.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getActiveFeatures()`

```
SubscriptionFeatureList getActiveFeatures(string usn)
```

This method retrieves all active features on a subscription for a specified USN.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

### Example: Request active features

This example shows a request for active features of subscription 2142420773.

```
<getActiveFeatures xmln="http://xml.inomial.com/smile/2.xsd">
  <usn>2142420773</usn>
</getActiveFeatures>
```

### Results

This method returns a SubscriptionFeatureList XML document that contains information about any features that have been applied to the subscription.

There are no messages defined by this method.

### Example: A returned SubscriptionFeatureList XML document

This example shows a SubscriptionFeatureList XML document. The document contains the properties of two features applied to a subscription.

```
<SubscriptionFeatureList xmln="http://xml.inomial.com/smile/2.xsd"
 xmln="http://ws.inomial.com/smile.2">
  <SubscriptionFeature>
    <Feature>
      <FeatureId>812</FeatureId>
      <Name>Test Feature 812</Name>
      <Object/>
    </Feature>
  </SubscriptionFeature>
  <SubscriptionFeature>
    <Feature>
      <FeatureId>813</FeatureId>
      <Name>Test Feature 813</Name>
      <Object/>
    </Feature>
  </SubscriptionFeature>
</SubscriptionFeatureList>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN specified does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetActiveFeatures usn
```

***usn***

**Purpose:** Specifies the USN of the subscription fo which to retrieve the active features.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage subscriptions

Managing subscriptions lets you change a subscription's data or plan, and suspend and resume the subscription.

The following methods manage subscriptions:

- `update()` updates subscription information.
- `changeUsername()` changes the username of a subscription.
- `setClearPassword()` changes the password for an account or subscription.
- `cancel()` closes a subscription on a given date.
- `suspend()` suspends a subscription's access to a service on a given date.
- `resume()` reinstates a subscription's access to a service on a given date.
- `applyFeature()` applies a feature to a subscription for a specified length of time.
- `applyFeatureToConnections()` applies a feature to any active connections of a USN.
- `stopApplyingFeature()` changes the date on which Smile stops applying a feature.
- `setCostCentre()` sets a cost centre for a subscription.
- `migrateSubscription()` migrates a subscription from one account to another nominated account.

### `update()`

```
Subscription update(string usn, Subscription subscription)
```

This method updates personal subscription data, for example, personal details and any custom fields you have added to the service.

This method updates elements nested within the `<Object>` element in a Subscription XML document, and may also update accessory data. You cannot use this method to update any elements that are outside the `<Object>` element. For example, you cannot use this method to update a subscription's status or parent. Nor can it be used to update billing fields such as invoicing cycle and rating cycle.

You can call this method at any point after the subscription order is entered.

**Note:** A username cannot be updated by this method. Use changeUsername() to change a username.

**Note:** For an example of a Subscription XML document, see the examples in SOAP envelopes.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to update.

**subscription**

**Format:** Subscription XML

**Purpose:** Defines a Subscription XML document that specifies the subscription data to update.

### Example: Request an update of subscription data

This example shows a Subscription XML document containing subscription data to be updated, including two custom fields.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <USN>1000000016</USN>
    <Properties>
      <Object>
        <String name="contactGiven">Leslie</String>
        <String name="contactFamily">Knope</String>
        <String name="emailAddress">leslie@pawneecouncil.gov</String>
        <String name="mobilePhone">0401555678</String>
        <Boolean name="specialCustomer">true</Boolean>
        <Decimal name="speed">123.51</Decimal>
      </Object>
    </Properties>
</Subscription>
```

### Results

This method returns a Subscription XML document that contains the updated state of the subscription. Smile also emits the `SubscriptionUpdated` message.

### Example: A returned Subscription XML document

This example shows a Subscription XML document with the complete updated state of the subscription.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <USN>1000000016</USN>
  <SID>17</SID>
  <ServiceName>Smile Access</ServiceName>
  <Parent>1000000008</Parent>
  <Properties>
    <Object>
      <String name="abn"></String>
      <String name="billAddress"></String>
      <String name="billBuildingName"></String>
      <String name="billCountry"></String>
      <String name="billFloorNumber"></String>
      <String name="billLotNumber"></String>
      <String name="billName"></String>
      <String name="billNumber"></String>
      <String name="billPostCode"></String>
      <String name="billPostalDeliveryType"></String>
      <String name="billState"></String>
      <String name="billSubUnit"></String>
      <String name="billSuburb"></String>
      <String name="billType"></String>
      <String name="comments"></String>
      <String name="companyPosition"></String>
      <String name="contactFamily">Knope</String>
      <String name="contactGiven">Leslie</String>
```

```
      <String name="contactTitle"></String>
      <String name="emailAddress">leslie@pawneecouncil.gov</String>
      <String name="fax"></String>
      <String name="homePhone"></String>
      <String name="jabberAddress"></String>
      <String name="mobilePhone">0401555678</String>
      <String name="name">Filiopoulos, Andrew</String>
      <String name="secretAnswer"></String>
      <String name="secretQuestion"></String>
      <String name="sex"></String>
      <String name="streetAddress"></String>
      <String name="streetBuildingName"></String>
      <String name="streetCountry"></String>
      <String name="streetFloorNumber"></String>
      <String name="streetLotNumber"></String>
      <String name="streetName"></String>
      <String name="streetNumber"></String>
      <String name="streetPostalDeliveryType"></String>
      <String name="streetPostcode"></String>
      <String name="streetState"></String>
      <String name="streetSubUnit"></String>
      <String name="streetSuburb"></String>
      <String name="streetType"></String>
      <String name="tradingName"></String>
      <String name="workPhone"></String>
      <String name="username">david@inomial.com</String>
      <String name="description"></String>
      <String name="filterId"></String>
      <String name="framedInterfaceId"></String>
      <String name="framedIpAddress"></String>
      <String name="framedIpNetmask"></String>
      <String name="framedIpv6Pool"></String>
      <String name="framedIpv6Route"></String>
      <String name="framedNetworkPrefix"></String>
      <String name="framedRoute"></String>
      <Integer name="preferredHomeContactTime"/>
      <Integer name="preferredMobileContactTime"/>
      <Integer name="preferredWorkContactTime"/>
      <Integer name="framedProtocol"/>
      <Integer name="radiusServiceType">1</Integer>
      <Boolean name="billPrintName">true</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
      <Boolean name="specialCustomer">true</Boolean>
      <Decimal name="speed">123.51</Decimal>
      <Timestamp name="dob"></Timestamp>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>NotActivated</ActivateStatus>
  <EnableStatus>NotEnabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>31</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
```

```
    <CycleDay>31</CycleDay>
  </RatingCycle>
  <Timezone key="13">Australia/Melbourne</Timezone>
  <Company key="1">Example Telecom</Company>
</Subscription>
```

### Faults

`UpdateException`

This fault is returned when the subscription does not exist, the caller does not have permission to update the subscription or the caller cannot modify a property.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateSubscription usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to update.

***filename***

**Purpose:** Defines the filename of the Subscription XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Related concepts

Example: Update a subscription on page 32

## `changeUsername()`

```
void changeUsername(string usn, string username)
```

This method changes the username of a subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to update.

**username**

**Format:** string

**Purpose:** Specifies the new username.

### Results

This method does not return a result for a successful update. An exception will be generated if the change attempt fails.

Smile emits the `SubscriptionUpdated` message after the username has been updated.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ChangeUsername usn username
```

***usn***

**Purpose:** Specifies the USN of the subscription to update.

**username**

**Purpose:** Specifies the new username.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `setClearPassword()`

```
void setClearPassword(string usn, Credential credential)
```

This method changes the password for an account or subscription. If an account has multiple subscriptions that use the same password, you must change the password for each subscription individually.

You must specify the password in clear text. After you change the password, Smile encrypts the password and stores it in the database.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account or subscription for which to change the password.

**credential**

**Format:** Credential XML

**Purpose:** Defines a Credential XML document that specifies the new password. The password cannot be encrypted.

### Example: Change a password

This example changes the password to Pa55w0rd.

```
<Credential>
  <Type>password</Type>
  <Parameters>
    <Object>
      <String name="password">Pa55w0rd</String>
    </Object>
  </Parameters>
</Credential>
```

### Results

This method does not return any output, and changes the password for the specified USN.

There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SubscriptionChangePassword usn password
```

***usn***

**Purpose:** Specifies the USN of the account or subscription for which to change the password.

***password***

**Purpose:** Specifies the new password.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `cancel()`

```
void cancel(string usn, date fromTimestamp)
```

This method closes a subscription at the specified date and time. If the specified time is before the current time, Smile closes the subscription immediately.

To close a subscription, Smile adds a plan schedule item with the close date to the plan schedule. This lets you calculate early termination fees or issue a credit for recurring charges in the next billing run.

After Smile closes a subscription, the subscription is deprovisioned and the subscription username is released and reset to the USN. Smile also releases any other resources held by the subscription.

If an account has multiple subscriptions, you must call this method for each subscription to close each subscription individually. You cannot close a subscription that is under contract; you must terminate the contract before you close the subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to cancel.

**fromTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time to close the subscription. Any usage after this time is unchargeable. For example, you cannot charge for any usage after the specified timestamp if your upstream provider does not immediately terminate the subscription.

### Results

This method does not return any output, and adds a plan schedule item with the close date to the plan schedule.

After the subscription is closed, Smile sends the `SubscriptionClosed` message to the message bus.

### Faults

`CancelException`

This fault is returned when the subscription cannot be cancelled, or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CancelSubscription usn
```

***usn***

**Purpose:** Specifies the USN of the subscription to immediately close. You cannot use the Java client to close a subscription on a future date.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `suspend()`

```
void suspend(string usn, date fromTimestamp, string comment)
```

This method suspends a subscription's access to a service at a specified date and time, and also suspends usage and recurring charges for the subscription, for example, monthly access fees.

If a suspended subscription is not under contract, any recurring charges will be pro-rated to exclude the suspension period. If a suspended subscription is under contract, the customer is obligated to pay any recurring charges for the duration of the contract; instead, recurring charges will be suspended after the contract ends, for the same period that the subscription was suspended.

You can call this method at any point in the subscription lifecycle after the subscription is activated and before it is closed. To suspend a subscription for a period of time in advance, call `suspend()` and `resume()` before the suspension date.

**Warning:** Suspending a subscription cancels any future plan changes.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to suspend.

**fromTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date from which the subscription is suspended. If you specify null, the subscription is suspended immediately.

**comment**

**Format:** string

**Purpose:** (Optional) Explains why the subscription was suspended. Smile stores the comment in the event log.

### Results

This method does not return any output. Smile suspends the subscription at the specified time.

Smile emits the `SubscriptionSuspended` message after the subscription is suspended.

### Faults

`InvalidRequestException`

This fault is returned when the specified USN is an account, or a subscription that cannot be suspended.

`NoSuchItemException`

This fault is returned when the specified USN is not an active subscription. An active subscription is provisioned and is not closed.

`SuspendException`

This fault is returned when the subscription cannot be suspended or the caller does not have permission to suspend the subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SuspendSubscription usn timestamp
 [comment]
```

***usn***

**Purpose:** Specifies the USN of the subscription to suspend.

***timestamp***

**Format:** dateTime

**Purpose:** Specifies the date from which the subscription is suspended.

***comment***

**Purpose:** (Optional) Explains why the subscription was suspended.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

**`resume()`**

```
void resume(string usn, date fromTimestamp, string comment)
```

This method reinstates a suspended subscription's access to a service and re-activates all usage charges and recurring charges.

You can call this method at any point in the subscription lifecycle after the subscription is activated and before it is closed. To suspend a subscription for a period of time in advance, call `suspend()` and `resume()` before the suspension date.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to resume.

**fromTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date from which the subscription is resumed.

**comment**

**Format:** string

**Purpose:** (Optional) Explains why the subscription was resumed. Smile stores the comment in the event log.

### Results

This method does not return any output. Smile resumes the subscription at the specified time.

Smile emits the `SubscriptionResumed` message after the subscription is resumed.

### Faults

`InvalidRequestException`

This fault is returned when the specified USN is an account, or a subscription that cannot be resumed.

`NoSuchItemException`

This fault is returned when the specified USN is not an active subscription. An active subscription is provisioned and is not closed.

`SuspendException`

This fault is returned when the subscription cannot be resumed, or the caller does not have permission to resume the subscription. You cannot resume a subscription that is not suspended.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ResumeSubscription usn timestamp
 [comment]
```

***usn***

**Purpose:** Specifies the USN of the subscription to resume.

***timestamp***

**Format:** dateTime

**Purpose:** Specifies the date from which the subscription is suspended.

***comment***

**Purpose:** (Optional) Explains why the subscription was suspended.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `applyFeature()`

```
subscriptionFeatureId applyFeature(string usn, int featureId,
 date startDate, date endDate)
```

This method applies a feature to a subscription for the specified length of time. A feature is an item that modifies a subscription and that can be switched on and off, for example, voicemail, call forwarding or DSL shaping.

The method returns a `subscriptionFeatureId` string that is unique to each call. Use this string to call `stopApplyingFeature()` and change the date that Smile stops applying the feature to the subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to apply the feature against.

**featureId**

**Format:** integer

**Purpose:** Specifies the ID of the feature to apply. The feature ID is located in the **Feature ID** field when you view the feature in Smile.

**startDate**

**Format:** date

**Purpose:** Specifies the date at which Smile applies the feature. Smile applies the feature at midnight at the start of the specified day.

If you specify null, Smile immediately applies the feature.

**endDate**

**Format:** date

**Purpose:** Specifies the date at which Smile stops applying the feature. Smile stops applying the feature at midnight at the start of the specified day.

If you specify null, Smile does not set an end date and applies the feature indefinitely.

### Results

This method applies the feature at the specified time and returns the `subscriptionFeatureId` parameter. Smile returns a different value for this parameter each time you call this method.

Smile emits the `SubscriptionFeaturesChanged` message when Smile applies the feature or stops applying the feature.

### Faults

`NoSuchItemException`

This fault is returned when the specified subscription or feature does not exist.

`InvalidRequestException`

This fault is returned when the end date is earlier than the start date.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ApplyFeature usn featureId
 [startDate] [endDate]
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to apply the feature.

***featureId***

**Purpose:** Specifies the ID of the feature to apply.

***startDate***

**Purpose:** (Optional) Specifies the date at which Smile applies the feature. If you do not specify this parameter, Smile applies the feature immediately.

***endDate***

**Purpose:** (Optional) Specifies the date at which Smile stops applying the feature. If you do not specify this parameter, Smile applies the feature indefinitely.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### applyFeatureToConnections()

```
void applyFeatureToConnections(string usn, int featureId)
```

This method applies a feature to any active connections for the given USN. The feature applies to each connection only for the duration of the connection. If the connection is stopped, subsequent connections will not be affected by this operation.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN whose connections will have the features applied.

**featureId**

**Format:** integer

**Purpose:** Specifies the numeric ID of the feature to apply to the connections.

#### Results

This method does not return any output.

Smile emits the `ConnectionFeaturesUpdated` message for each connection whose features change due to this operation.

#### Faults

`NoSuchItemException`

This fault is returned when both or either the provided USN or the provided featureId do not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ApplyFeatureToConnections usn featureId
```

***usn***

**Purpose:** Specifies the USN whose connections will have the features applied.

***featureId***

**Purpose:** Specifies the numeric ID of the feature to apply to the connections.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### stopApplyingFeature()

```
void stopApplyingFeature(string subscriptionFeatureId, date endDate)
```

This method changes or removes the date that Smile stops applying a feature to a subscription.

Use this method to change the length of time the feature is applied to the subscription, or to apply the feature to the subscription indefinitely.

#### Parameters

**subscriptionFeatureId**

**Format:** string

**Purpose:** Specifies the string returned by the `applyFeature()` method that uniquely identifies the subscription, feature and length of time the feature is applied.

**Note:** Smile returns a different `subscriptionFeatureId` each time you call `applyFeature()`.

**endDate**

**Format:** date

**Purpose:** Specifies the date to stop applying the feature. Smile stops applying the feature at midnight at the start of the specified day.

If you specify null, Smile removes the end date and applies the feature indefinitely.

#### Results

This method does not return any output, and changes the date that Smile stops applying the feature.

Smile emits the `SubscriptionFeaturesChanged` message when Smile stops applying the feature.

#### Faults

`NoSuchItemException`

This fault is returned when the specified `subscriptionFeatureId` does not exist.

`InvalidRequestException`

This fault is returned when the `subscriptionFeatureId` is not specified.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.StopApplyingFeature
 subscriptionFeatureId [endDate]
```

***subscriptionFeatureId***

**Purpose:** Specifies the ID of the feature for which to specify an end date.

***endDate***

**Purpose:** (Optional) Specifies the date to stop applying the feature.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `setCostCentre()`

```
void setCostCentre(string usn, int costCentreId)
```

This method sets the cost centre given by the CostCentreId field to a subscription. When no cost centre ID is provided, it will clear the subscription's cost centre.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the subscription USN.

**costCentreId**

**Format:** integer

**Purpose:** Specifies the ID of the cost centre.

#### Results

This method does not return any output, and sets a cost centre for a subscription.

Smile emits the `RatingConfigurationUpdated` message when the task is complete.

#### Faults

`NoSuchItemException`

This fault is returned when:

- a cost centre with the specified ID does not exist
- a subscription with the specified USN does not exist

`InvalidRequestException`

This fault is returned when the cost centre does not belong to the account of the given subscription.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SetCostCentre usn costCentreId
```

***usn***

**Purpose:** Specifies the subscription USN.

***costCentreId***

**Purpose:** Specifies the ID of the cost centre.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `migrateSubscription()`

```
void migrateSubscription(string usn, string toAccountUsn)
```

This method migrates a subscription from one account to another nominated account. These two accounts must have the same currency.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to migrate.

**toAccountUsn**

**Format:** string

**Purpose:** Specifies the USN of the account that the subscription is migrating to.

#### Results

This method does not return any output.

There are no messages defined by this method.

#### Faults

`InvalidRequestException`

This fault is returned when:

- an invalid USN is provided
- the current account has a different currency from the account that the subscription is migrating to
- the destination account has commissions that require a contract

`NoSuchItemException`

This fault is returned when either of the specified USNs do not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.MigrateSubscription usn
 toAccountUsn
```

***usn***

**Purpose:** Specifies the USN of the subscription to migrate.

***toAccountUsn***

**Purpose:** Specifies the USN of the account that the subscription is migrating to.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage plans

Managing plans lets you retrieve information about a subscription's plan schedule or change a subscription's plan.

The plan schedule specifies the plans that a subscription uses throughout its lifecycle. The plan schedule is made up of plan schedule items, which specify when a plan becomes active for the subscription.

The following methods manage plans:

- `getPlanSchedules()` returns information about all plan schedules associated with a subscription.
- `getPlanScheduleItemsAt()` returns information about the plan a subscription uses at a specified time.
- `getPlanChargeDescription()` returns information about the charge configuration for a subscription according to a given plan schedule item.
- `changePlan()` changes the plan for a subscription.

### `getPlanSchedules()`

```
PlanSchedules getPlanSchedules(string usn)
```

This method returns information about each plan a subscription uses throughout its lifecycle. Each plan is described by a plan schedule, which specifies the plan that a subscription uses at a given point in time.

Use this method to retrieve information about a subscription's current plan, plan changes scheduled in the future and plan changes that occurred in the past.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the plans.

#### Results

This method returns a PlanSchedules XML document that contains information about each plan the subscription uses throughout its lifecycle. The plans are defined by plan schedule items, which specify when the subscription begins using each plan. The PlanSchedules XML document lists the plan schedule items in chronological order.

There are no messages defined by this method.

#### Example: A returned PlanSchedules XML document

This example shows a PlanSchedules XML document that contains two plan schedule items. These items show that the subscription began using the Silver plan on 1 January 2011, and then used the Gold plan from 1 July 2011.

```
<PlanSchedules xmlns="http://xml.inomial.com/smile/2.xsd">
  <PlanSchedule>
    <PlanScheduleType>1</PlanScheduleType>
```

```
    <PlanScheduleItem>
      <PlanCode>Silver</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2011-01-01T00:00:00+11:00</Start>
      <Options>
        <Object>
          <Boolean name="staticIp">true</Boolean>
        </Object>
      </Options>
    </PlanScheduleItem>
    <PlanScheduleItem>
      <PlanCode>Gold</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2011-07-01T00:00:00+11:00</Start>
      <Options>
        <Object>
          <Boolean name="staticIp">true</Boolean>
        </Object>
      </Options>
    </PlanScheduleItem>
  </PlanSchedule>
</PlanSchedules>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPlanSchedules usn
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the plans.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getPlanScheduleItemsAt()`

```
PlanScheduleItems getPlanScheduleItemsAt(string usn, dateTime
 timestamp)
```

This method returns information about the plan a subscription uses at a specified time.

Use this method to retrieve and display information about a subscription's plan to a customer.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the plan.

**timestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time for which you want to retrieve plan information.

### Results

This method returns a PlanScheduleItems XML document that contains information about the plan or plans the subscription uses at the specified time. If you specify a timestamp before the subscription was created, Smile returns an empty response.

There are no messages defined by this method.

### Example: A returned PlanScheduleItems XML document

This example shows a PlanScheduleItems XML document that lists the plan schedule items associated with the subscription on 1 January 2011. The two plan schedule items are of different types.

Associating more than one type of plan schedule with a subscription lets you rate CDRs in different ways, for example, charging for incoming and outgoing calls.

**Note:** This example shows an advanced use case. Generally, only one plan schedule type is associated with a subscription, and the value of the `<PlanScheduleType>` element is usually 1.

```
<PlanScheduleItems xmlns="http://xml.inomial.com/smile/2.xsd">
  <PlanScheduleItem>
    <PlanCode>Silver</PlanCode>
    <PlanScheduleType>1</PlanScheduleType>
    <Start>2011-01-01T00:00:00+11:00</Start>
    <Options>
      <Object>
        <Boolean name="staticIp">true</Boolean>
      </Object>
    </Options>
  </PlanScheduleItem>
  <PlanScheduleItem>
    <PlanCode>Value1</PlanCode>
    <PlanScheduleType>3</PlanScheduleType>
    <Start>2011-01-01T00:00:00+11:00</Start>
    <Options>
      <Object/>
    </Options>
  </PlanScheduleItem>
</PlanScheduleItems>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

`InvalidRequestException`

This fault is returned when the caller does not specify a timestamp.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPlanScheduleItemsAt usn timestamp
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the plan.

***timestamp***

**Purpose:** Specifies the date and time for which you want to retrieve plan information.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getPlanChargeDescription()`

```
PlanChargeDescriptionType getPlanChargeDescription(string usn,
 integer planScheduleItem)
```

This method returns a document describing the charge configuration for the subscription, according to the given plan schedule item.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription or account.

**planScheduleItem**

**Format:** integer

**Purpose:** Specifies a plan schedule item to also consult.

### Results

This method returns a ConnectionListType XML document that contains information about the charge configuration for the subscription, according to the given plan schedule item.

The ConnectionListType XML document has a number of parts:

- `planScheduleItem`—details of the plan schedule item
- `planConfiguration` - charging details of the plan. It includes both recurring charges, and usage charging rules.
- `planScheduleItemOverrides`—the overrides to the plan configuration from the plan schedule item. Any non-empty fields are considered to be overrides for the values in the plan configuration.

There are no messages defined by this method.

### Example: A returned ConnectionListType XML document

This example shows a ConnectionListType XML document with two connection historys and has no more existing results.

```
<PlanChargeDescription xmlns="http://xml.inomial.com/smile/2.xsd">
```

```
  <PlanScheduleItem>
    <PlanCode>tp2841</PlanCode>
    <PlanScheduleType>1</PlanScheduleType>
    <Start>2013-05-10T00:00:00.000+10:00</Start>
    <Options>
      <Object>
        <Integer name="option1">49</Integer>
        <Integer name="option2">1</Integer>
      </Object>
    </Options>
    <PlanScheduleItem>417</PlanScheduleItem>
  </PlanScheduleItem>
  <PlanConfiguration>
    <RecurringCharges>
      <RecurringCharge id="223">
        <ChargeQuantity>1</ChargeQuantity>
        <Frequency type="Rating Periods">1</Frequency>
        <WhenToCharge>In Advance</WhenToCharge>
        <AdvancePeriods>1</AdvancePeriods>
        <ProRataAction>Pro-Rata</ProRataAction>
        <MinProRataDays>1</MinProRataDays>
        <LineItemDescription>
          <ItemCode>002307</ItemCode>
          <Name>Invoice item description</Name>
          <UnitSize>1</UnitSize>
          <ChargeGst>true</ChargeGst>
          <Rate currency="AUD" formatted="$100.00">100</Rate>
          <DisplayRate>$100.00 excludes GST</DisplayRate>
        </LineItemDescription>
        <ValidFrom>2013-05-10+10:00</ValidFrom>
      </RecurringCharge>
      <RecurringCharge id="224">
        <ChargeQuantity>1</ChargeQuantity>
        <Frequency type="Rating Periods">1</Frequency>
        <WhenToCharge>In Advance</WhenToCharge>
        <AdvancePeriods>1</AdvancePeriods>
        <ProRataAction>Pro-Rata</ProRataAction>
        <MinProRataDays>1</MinProRataDays>
        <LineItemDescription>
          <ItemCode>002308</ItemCode>
          <Name>Invoice item description</Name>
          <UnitSize>1</UnitSize>
          <ChargeGst>true</ChargeGst>
          <Rate currency="AUD" formatted="$90.00">90</Rate>
        </LineItemDescription>
      </RecurringCharge>
    </RecurringCharges>
    <PlanOptions>
      <PlanOption option="option1">
        <Name>Option 1</Name>
        <DisplayType>NUMERIC</DisplayType>
        <MinimumQuantity>1</MinimumQuantity>
        <MaximumQuantity>100</MaximumQuantity>
        <RecurringCharges>
          <RecurringCharge id="225">
            <ChargeQuantity>1</ChargeQuantity>
            <Frequency type="Rating Periods">1</Frequency>
```

```
            <WhenToCharge>In Advance</WhenToCharge>
            <AdvancePeriods>1</AdvancePeriods>
            <ProRataAction>Pro-Rata</ProRataAction>
            <MinProRataDays>1</MinProRataDays>
            <LineItemDescription>
              <ItemCode>002309</ItemCode>
              <Name>Invoice item description</Name>
              <UnitSize>1</UnitSize>
              <ChargeGst>true</ChargeGst>
              <Rate currency="AUD" formatted="$25.60">25.6</Rate>
            </LineItemDescription>
          </RecurringCharge>
        </RecurringCharges>
      </PlanOption>
      <PlanOption option="option2">
        <Name>Option 2</Name>
        <DisplayType>BOOLEAN</DisplayType>
        <Group key="223">testOptionGroup1</Group>
        <RecurringCharges/>
      </PlanOption>
    </PlanOptions>
    <ChargingRules>
      <ChargingRule id="1040">
        <ChargeType key="0">Online time</ChargeType>
        <LineItemDescription>
          <ItemCode>002310</ItemCode>
          <Name>This goes on the invoice</Name>
          <UnitSize>1</UnitSize>
          <ChargeGst>true</ChargeGst>
          <Rate currency="AUD" formatted="$9.50">9.5</Rate>
          <DisplayRate>$9.5 per unit</DisplayRate>
        </LineItemDescription>
        <RateSource>
          <FixedCharge>
            <Rate currency="AUD" formatted="$9.50">9.5</Rate>
            <PerUnit unitQuantity="1">second</PerUnit>
          </FixedCharge>
        </RateSource>
        <Constraints>
          <DaysOfWeek>
            <DayOfWeek>Friday</DayOfWeek>
          </DaysOfWeek>
          <FromDate>2013-05-10+10:00</FromDate>
          <ToDate>2013-05-12+10:00</ToDate>
          <FromTime>00:00:00.000+10:00</FromTime>
          <ToTime>23:59:59.000+10:00</ToTime>
          <PlanOption key="223">Option 1</PlanOption>
          <MaximumUnits>5000.000</MaximumUnits>
          <MaximumUnitsPerDayOptionMultiplier>2</
MaximumUnitsPerDayOptionMultiplier>
          <MaximumUnitsPerSession>3000.000</MaximumUnitsPerSession>
          <MaximumSessionLengthSeconds>1000000</
MaximumSessionLengthSeconds>
        </Constraints>
        <PrepaidRules/>
        <AppliesFeatures>
          <Feature key="224">Test Feature 224</Feature>
```

```
        </AppliesFeatures>
        <IncludedValuePoolRef key="813">Cap</IncludedValuePoolRef>
      </ChargingRule>
      <ChargingRule id="1041">
        <ChargeType key="1">Downloaded Data</ChargeType>
        <LineItemDescription>
          <ItemCode>002311</ItemCode>
          <Name>Downloaded Data</Name>
          <UnitSize>1</UnitSize>
          <ChargeGst>true</ChargeGst>
          <Rate currency="AUD" formatted="$0.00">0</Rate>
        </LineItemDescription>
        <RateSource>
          <Tariff>Rate</Tariff>
        </RateSource>
        <Constraints>
          <DaysOfWeek all="true"/>
        </Constraints>
        <PrepaidRules/>
      </ChargingRule>
    </ChargingRules>
    <IncludedValuePools>
      <IncludedValuePool id="813">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
  <PlanScheduleItemOverrides>
    <RecurringCharges>
      <RecurringCharge id="223">
        <Rate currency="AUD" formatted="$34.00">34</Rate>
        <Comment>This is override invoice comment</Comment>
        <ChargeQuantity>19</ChargeQuantity>
        <Description>overrides description</Description>
        <Frequency type="Rating Periods">1</Frequency>
        <WhenToCharge>In Advance</WhenToCharge>
        <AdvancePeriods>2</AdvancePeriods>
      </RecurringCharge>
    </RecurringCharges>
    <ChargingRules/>
  </PlanScheduleItemOverrides>
</PlanChargeDescription>
```


### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription, or the specified plan schedule item does not exist

`InvalidRequestException`

This fault is returned when the specified plan schedule item does not match the specified account or subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPlanChargeDescription usn
 planScheduleItem
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the plans.

***planScheduleItem***

**Purpose:** Specifies a plan schedule item to also consult.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `changePlan()`

```
void changePlan(string usn, PlanScheduleItem PlanScheduleItem)
```

This method adds an item to the subscription's plan schedule. The plan schedule specifies the plan that a subscription uses a given point in time. You can add an item that changes the subscription's plan now, in the past or at a future point, for example, when a contract ends.

You cannot use this method to add plan option groups or view the subscription's current plan schedule. To add a plan options group, change the plan schedule in Smile.

You can call this method at any point after the subscription is activated; however, you may not be able to change the plan of a subscription that is under contract. If you suspend a subscription, Smile cancels any future plan changes.

**Warning:** Do not call this method on a closed or suspended subscription. This method overrides any other scheduled plan changes, except close changes.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to change the plan.

**PlanScheduleItem**

**Format:** PlanScheduleItem XML

**Purpose:** Defines a PlanScheduleItem XML document that contains the item to add to the plan schedule.

### Example: Change a plan

This example shows a PlanScheduleItem XML document that adds the Silver plan with a static IP option to a subscription's plan schedule and specifies the number of modem ports. The plan takes effect 01/01/2013. The plan schedule item also overrides the flagfall for China cellular calls and overrides some charging rules with new limits.

```
<PlanScheduleItem xmlns="http://xml.inomial.com/smile/2.">
  <PlanCode>Silver</PlanCode>
```

```
  <PlanScheduleType>1</PlanScheduleType>
  <Start>2013-01-01T00:00:00+11:00</Start>
  <Options>
    <Object>
      <Boolean name="staticIp">true</Integer>
      <Integer name="modemPorts">6</Integer>
    </Object>
  </Options>
  <TariffOverrides>
    <TariffOverride>
      <Destination key="69751">China cellular</Destination>
      <TariffType key="1541">Local</TariffType>
      <Flagfall>2</Flagfall>
      <Rate>1</Rate>
      <Units>1</Units>
      <Increment>1</Increment>
      <MinimumCharge>5</MinimumCharge>
      <MaximumCharge>20</MaximumCharge>
      <OverrideDescription>Updated from API</OverrideDescription>
      <RoundingMode>HalfUp</RoundingMode>
      <DecimalPlaces>2</DecimalPlaces>
      <CallType>Voice</CallType>
    </TariffOverride>
  </TariffOverrides>
  <ChargingRuleOverrides>
    <ChargingRuleOverride>
      <ChargingRule key="1119" />
      <Rate>1</Rate>
      <MaximumUnits>10</MaximumUnits>
      <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
      <Comment>SOAP</Comment>
    </ChargingRuleOverride>
    <ChargingRuleOverride>
      <ChargingRule key="1118" />
      <RetailMarkup>50</RetailMarkup>
      <MaximumUnits>10</MaximumUnits>
      <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
      <Comment>SOAP</Comment>
    </ChargingRuleOverride>
  </ChargingRuleOverrides>
</PlanScheduleItem>
```

### Results

This method does not return any output, and adds a plan schedule item to the subscription's plan schedule.

Smile does not emit any messages when the plan changes, but if the plan change causes the subscription's features to change, Smile emits the `SubscriptionFeaturesChanged` message when the feature change is complete.

### Faults

`ChangePlanException`

This fault is returned when a required plan option is not specified or the plan cannot be changed, for example, the plan is a contract.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ChangePlan usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to update.

***filename***

**Purpose:** Defines the filename of the PlanScheduleItem XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage real-time rating plan changes

Managing real-time rating plans lets you retrieve available plan information for services and subscriptions, and propose and commit plan changes. The plan change endpoint is only available for BigRating (RTR) subscriptions.

The plan change WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/subscription/planchange?wsdl`.

The following methods manage real-time rating plan changes:

- `getSubscriptionPlans()` returns a list of published plans available to a subscription.
- `getServicePlans()` returns a list of published plans available against a service or plans available to subscriptions of accounts of a given service.
- `proposePlanChange()` prepares a quote for a plan change.
- `commitPlanChange()` performs a plan change based on a previously acquired PlanChangeOffer.

### `getSubscriptionPlans()`

```
PlanConfigurations getSubscriptionPlans(string usn, date timestamp)
```

This method returns a list of published plans available to a subscription. Available plans may also be listed at a specific date and time.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to return available plans.

**timestamp**

**Format:** date

**Purpose:** (Optional) Specifies a specific point in time at which to list available plans.

#### Results

This method returns a list of published plans available for the subscription. The private plans of an account that this subscription belongs to will be included in the returned list. If you do not specify a date and time, it will return a list of valid plans at the time the request is sent.

There are no messages defined by this method.

#### Example: A returned PlanConfigurationsType XML document

This example shows a PlanConfigurationsType XML document when subscription USN and, optional, date and time are given as parameters. Each `<PlanConfiguration>` element contains information about the plan.

```
<PlanConfigurations xmlns:ns2="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
```

```
  <PlanConfiguration>
    <Plan>tp858_1</Plan>
    <PlanName>Test plan 858 _1</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2455">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
  <PlanConfiguration>
    <Plan>tp858_2</Plan>
    <PlanName>Test plan 858 _2</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2457">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
  <PlanConfiguration>
    <Plan>tpp859</Plan>
    <PlanName>Test private plan 859</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2467">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
</PlanConfigurations>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match a subscription.

`InvalidRequestException`

This fault is returned when the subscription USN is not provided as a parameter.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionPlans usn
 [timestamp]
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to return available plans.

***timestamp***

**Purpose:** (Optional) Specifies a specific point in time at which to list available plans.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getServicePlans()`

```
PlanConfigurations getServicePlans(int sid, string usn, date
 timestamp)
```

This method returns a list of published plans available against a service or plans available to subscriptions on an accounts for a service. Available plans may also be listed at a specific date and time.

### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID for which to return available plans.

**usn**

**Format:** string

**Purpose:** (Optional) Specifies the USN of the account for which to return available plans.

**timestamp**

**Format:** date

**Purpose:** (Optional) Specifies a specific point in time at which to return available plans.

### Results

This method returns a list of published plans available against a service or plans available to subscriptions on an account for a service. The private plans of an account for the specified service will be included in the list when account USN is specified. If you do not specify a date and time, it will return a list of valid plans at the time the request is sent.

There are no messages defined by this method.

### Example: A returned PlanConfigurationsType XML document

This example shows a PlanConfigurationsType XML document when service ID, and optional account USN and date and time are given as parameters. Each `<PlanConfiguration>` element contains information about the plan.

```
<PlanConfigurations xmlns:ns2="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <PlanConfiguration>
    <Plan>tp858_1</Plan>
    <PlanName>Test plan 858 _1</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2455">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
  <PlanConfiguration>
    <Plan>tp858_2</Plan>
    <PlanName>Test plan 858 _2</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2457">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
  <PlanConfiguration>
    <Plan>tpp859</Plan>
    <PlanName>Test private plan 859</PlanName>
    <Company>1</Company>
    <RecurringCharges/>
    <PlanOptions/>
    <ChargingRules/>
    <IncludedValuePools>
      <IncludedValuePool id="2467">
        <Name>Cap</Name>
        <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
      </IncludedValuePool>
    </IncludedValuePools>
  </PlanConfiguration>
</PlanConfigurations>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match a subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetServicePlans sid [usn]
 [timestamp]
```

***sid***

**Purpose:** Specifies the service ID for which to return available plans.

***usn***

**Purpose:** (Optional) Specifies the USN of the account for which to return available plans.

***timestamp***

**Purpose:** (Optional) Specifies a specific point in time at which to return available plans.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `proposePlanChange()`

```
PlanChangeOffer proposePlanChange(string usn, PlanChangeProposal
 planChangeProposal)
```

This method prepares a quote for a plan change. This quote may be committed to via the `commitPlanChange()` method.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to which the plan change is proposed.

**planChangeProposal**

**Format:** PlanChangeProposal XML

**Purpose:** Defines a PlanChangeProposal XML document that specifies the item for the plan change.

### Example: Propose a plan change

This example shows a PlanChangeProposal XML document that proposes the addition of plan2, with a quantity of two option 1s. This plan change would take effect 16/04/2014. The plan schedule item also overrides the flagfall for China cellular calls and overrides some charging rules with new limits.

```
<PlanChangeProposal xmlns="http://xml.inomial.com/smile/2.xsd">
  <PlanScheduleItem>
    <PlanCode>plan2</PlanCode>
    <PlanScheduleType>1</PlanScheduleType>
    <Start>2014-04-16T00:00:00</Start>
```

```
    <Options>
      <Object>
        <Integer name="op1">2</Integer>
      </Object>
    </Options>
    <TariffOverrides>
      <TariffOverride>
        <Destination key="69751">China cellular</Destination>
        <TariffType key="1541">Local</TariffType>
        <Flagfall>2</Flagfall>
        <Rate>1</Rate>
        <Units>1</Units>
        <Increment>1</Increment>
        <MinimumCharge>5</MinimumCharge>
        <MaximumCharge>20</MaximumCharge>
        <OverrideDescription>Updated from API</OverrideDescription>
        <RoundingMode>HalfUp</RoundingMode>
        <DecimalPlaces>2</DecimalPlaces>
        <CallType>Voice</CallType>
      </TariffOverride>
    </TariffOverrides>
    <ChargingRuleOverrides>
      <ChargingRuleOverride>
        <ChargingRule key="1119" />
        <Rate>1</Rate>
        <MaximumUnits>10</MaximumUnits>
        <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
        <Comment>SOAP</Comment>
      </ChargingRuleOverride>
      <ChargingRuleOverride>
        <ChargingRule key="1118" />
        <RetailMarkup>50</RetailMarkup>
        <MaximumUnits>10</MaximumUnits>
        <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
        <Comment>SOAP</Comment>
      </ChargingRuleOverride>
    </ChargingRuleOverrides>
  </PlanScheduleItem>
</PlanChangeProposal>
```

### Results

This method returns a PlanChangeOffer XML document that contains information about what the cost of the plan change would be.

The PlanChangeOffer also contains an `<OfferId>` which is used to determine if the offer is still valid. An offer is valid for as long as the information that was generated remains unchanged. For example, if a subscription has its plan changed, then any offers created before the plan change will become invalid.

There are no messages defined by this method.

### Example: A returned PlanChangeOffer XML document

This example shows a PlanChangeOffer XML document that contains information about the cost of the new plan and options and a credit for the subscription's current plan.

```
<PlanChangeOffer xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <USN>2142421144</USN>
  <PlanChangeProposal>
    <PlanScheduleItem>
      <PlanCode>plan2</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2014-04-16T00:00:00</Start>
      <Options>
        <Object>
          <Integer name="op1">2</Integer>
        </Object>
      </Options>
      <TariffOverrides>
        <TariffOverride>
          <Destination key="69751">China cellular</Destination>
          <TariffType key="1541">Local</TariffType>
          <Flagfall>2</Flagfall>
          <Rate>1</Rate>
          <Units>1</Units>
          <Increment>1</Increment>
          <MinimumCharge>5</MinimumCharge>
          <MaximumCharge>20</MaximumCharge>
          <OverrideDescription>Updated from API</OverrideDescription>
          <RoundingMode>HalfUp</RoundingMode>
          <DecimalPlaces>2</DecimalPlaces>
          <CallType>Voice</CallType>
        </TariffOverride>
      </TariffOverrides>
      <ChargingRuleOverrides>
        <ChargingRuleOverride>
          <ChargingRule key="1119" />
          <Rate>1</Rate>
          <MaximumUnits>10</MaximumUnits>
          <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
          <Comment>SOAP</Comment>
        </ChargingRuleOverride>
        <ChargingRuleOverride>
          <ChargingRule key="1118" />
          <RetailMarkup>50</RetailMarkup>
          <MaximumUnits>10</MaximumUnits>
          <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
          <Comment>SOAP</Comment>
        </ChargingRuleOverride>
      </ChargingRuleOverrides>
    </PlanScheduleItem>
    <OfferId>e34ab133-f570-3779-8695-3387ea20d21d</OfferId>
  </PlanChangeProposal>
  <NewInvoiceRequest>
    <effectiveDate>2014-04-16+10:00</effectiveDate>
    <dueDate>2014-04-28+10:00</dueDate>
```

```
    <ChargeRequest>
      <USN>2142421144</USN>
      <itemCode>001336</itemCode>
      <description>Plan 2 (16/04/2014 - 30/04/2014)</description>
      <quantity>1</quantity>
      <startDate>2014-04-16+10:00</startDate>
      <endDate>2014-05-01+10:00</endDate>
      <amount>7.49</amount>
      <chargeGst>true</chargeGst>
      <gst gstId="2">
        <amount>0.75</amount>
      </gst>
      <chargeId>c33d170a-855d-3b3e-a2dd-3b3cbea01c42</chargeId>
    </ChargeRequest>
    <ChargeRequest>
      <USN>2142421144</USN>
      <itemCode>001337</itemCode>
      <description>Option 1 (16/04/2014 - 30/04/2014)</description>
      <quantity>2</quantity>
      <startDate>2014-04-16+10:00</startDate>
      <endDate>2014-05-01+10:00</endDate>
      <amount>1.00</amount>
      <chargeGst>true</chargeGst>
      <gst gstId="2">
        <amount>0.10</amount>
      </gst>
      <chargeId>01e85a99-8e2e-3a9e-85e0-87b7a2d40f3b</chargeId>
     </ChargeRequest>
     <ChargeRequest>
       <USN>2142421144</USN>
       <itemCode>001335</itemCode>
       <description>Retraction for Plan 1 (16/04/2014 - 30/04/2014)</
description>
       <quantity>1.000</quantity>
       <startDate>2014-04-16+10:00</startDate>
       <endDate>2014-05-01+10:00</endDate>
       <amount>-4.99</amount>
       <chargeGst>true</chargeGst>
       <gst gstId="2">
         <amount>-0.50</amount>
       </gst>
       <chargeId>8ccb6f3f-4a5d-35bc-a41f-0d7a3280a269</chargeId>
     <retractChargeId>afd075b6-b7af-4438-99ef-848a8c9c890c</
retractChargeId>
    </ChargeRequest>
  </NewInvoiceRequest>
</PlanChangeOffer>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription, or another referenced object, does not exist.

`InvalidRequestException`

This fault is returned if the plan change is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ProposePlanChange usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to which the plan change is proposed.

***filename***

**Purpose:** Defines the filename of the PlanChangeProposal XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `commitPlanChange()`

```
PlanChangeResponse commitPlanChange(string usn, PlanChangeOffer
 planChangeOffer)
```

This method performs a plan change based on a previously acquired PlanChangeOffer.

Any changes made to the offer will be honoured by this method, however charges are identified by their `chargeId`. Any charges not present in the quote will be raised as deferred charges. This means that if you want to cancel a charge, you must leave it in the quote and set its value to $0.

Some fields such as GST may be removed in order to have Smile calculate charges.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription of the plan to be changed.

**planChangeOffer**

**Format:** PlanChangeOffer XML

**Purpose:** Defines a PlanChangeOffer XML document that defines the plan change and any charges that should be raised as a result of the plan change.

### Example: A PlanChangeOffer XML document

This example shows a PlanChangeOffer XML document that contains information about the cost of the new plan and options and a credit for the subscription's current plan. The plan schedule item also overrides the flagfall for China cellular calls and overrides some charging rules with new limits.

```
<PlanChangeOffer xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <USN>2142421144</USN>
  <PlanChangeProposal>
    <PlanScheduleItem>
      <PlanCode>plan2</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2014-04-16T00:00:00</Start>
      <Options>
        <Object>
```

```
          <Integer name="op1">2</Integer>
        </Object>
      </Options>
      <TariffOverrides>
        <TariffOverride>
          <Destination key="69751">China cellular</Destination>
          <TariffType key="1541">Local</TariffType>
          <Flagfall>2</Flagfall>
          <Rate>1</Rate>
          <Units>1</Units>
          <Increment>1</Increment>
          <MinimumCharge>5</MinimumCharge>
          <MaximumCharge>20</MaximumCharge>
          <OverrideDescription>Updated from API</OverrideDescription>
          <RoundingMode>HalfUp</RoundingMode>
          <DecimalPlaces>2</DecimalPlaces>
          <CallType>Voice</CallType>
        </TariffOverride>
      </TariffOverrides>
      <ChargingRuleOverrides>
        <ChargingRuleOverride>
          <ChargingRule key="1119" />
          <Rate>1</Rate>
          <MaximumUnits>10</MaximumUnits>
          <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
          <Comment>SOAP</Comment>
        </ChargingRuleOverride>
        <ChargingRuleOverride>
          <ChargingRule key="1118" />
          <RetailMarkup>50</RetailMarkup>
          <MaximumUnits>10</MaximumUnits>
          <MaximumUnitsPerSession>100</MaximumUnitsPerSession>
          <Comment>SOAP</Comment>
        </ChargingRuleOverride>
      </ChargingRuleOverrides>
    </PlanScheduleItem>
    <OfferId>e34ab133-f570-3779-8695-3387ea20d21d</OfferId>
  </PlanChangeProposal>
  <NewInvoiceRequest>
    <effectiveDate>2014-04-16+10:00</effectiveDate>
    <dueDate>2014-04-28+10:00</dueDate>
    <ChargeRequest>
      <USN>2142421144</USN>
      <itemCode>001336</itemCode>
      <description>Plan 2 (16/04/2014 - 30/04/2014)</description>
      <quantity>1</quantity>
      <startDate>2014-04-16+10:00</startDate>
      <endDate>2014-05-01+10:00</endDate>
      <amount>7.49</amount>
      <chargeGst>true</chargeGst>
      <gst gstId="2">
        <amount>0.75</amount>
      </gst>
      <chargeId>c33d170a-855d-3b3e-a2dd-3b3cbea01c42</chargeId>
    </ChargeRequest>
    <ChargeRequest>
      <USN>2142421144</USN>
```

```
      <itemCode>001337</itemCode>
      <description>Option 1 (16/04/2014 - 30/04/2014)</description>
      <quantity>2</quantity>
      <startDate>2014-04-16+10:00</startDate>
      <endDate>2014-05-01+10:00</endDate>
      <amount>1.00</amount>
      <chargeGst>true</chargeGst>
      <gst gstId="2">
        <amount>0.10</amount>
      </gst>
      <chargeId>01e85a99-8e2e-3a9e-85e0-87b7a2d40f3b</chargeId>
    </ChargeRequest>
    <ChargeRequest>
      <USN>2142421144</USN>
      <itemCode>001335</itemCode>
      <description>Retraction for Plan 1 (16/04/2014 - 30/04/2014)</
description>
      <quantity>1.000</quantity>
      <startDate>2014-04-16+10:00</startDate>
      <endDate>2014-05-01+10:00</endDate>
      <amount>-4.99</amount>
      <chargeGst>true</chargeGst>
      <gst gstId="2">
        <amount>-0.50</amount>
      </gst>
      <chargeId>8ccb6f3f-4a5d-35bc-a41f-0d7a3280a269</chargeId>
      <retractChargeId>afd075b6-b7af-4438-99ef-848a8c9c890c</
retractChargeId>
    </ChargeRequest>
  </NewInvoiceRequest>
</PlanChangeOffer>
```

### Results

This method returns a PlanChangeReponse XML document containing information for the invoice resulting from the plan change.

There are no messages defined by this method.

### Example: A returned PlanChangeResponse XML document

This example shows a returned PlanChangeResponse XML document. It shows that some of the fields were modified, including the GST value for the "Retraction for Plan 1", so that Smile's rules (10% GST) are not used.

```
<PlanChangeResponse xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <Invoice>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>21460407</transactionNumber>
    <currency>AUD</currency>
    <amount>9.15</amount>
```

```
    <accountType>1</accountType>
    <gstAmount>1.65</gstAmount>
    <entryTimestamp>2014-04-15T10:58:32.454+10:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$9.15">9.15</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$1.65">1.65</
formattedGstAmount>
    <unallocatedAmount>9.15</unallocatedAmount>
    <enteredByUsn>2142420807</enteredByUsn>
    <logTimestamp>2014-04-16T10:58:32.456+10:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$9.15">9.15</formattedUnallocatedAmount>
    <transactionDate>2014-04-15+10:00</transactionDate>
    <dueDate>2014-04-30+10:00</dueDate>
    <openDate>2014-04-16+10:00</openDate>
    <closeDate>2014-04-16+10:00</closeDate>
    <quoteNumber>21460407</quoteNumber>
    <createdByUsn>2142420807</createdByUsn>
    <discount>0.00</discount>
    <transactionSource>Manual</transactionSource>
    <transactionItem>
      <usn>1000000008</usn>
      <lineNumber>0</lineNumber>
      <itemCode>001337</itemCode>
      <chargeFrom>2014-04-16+10:00</chargeFrom>
      <chargeTo>2014-05-01+10:00</chargeTo>
      <description>Option 1 (16/04/2014 - 30/04/2014)</description>
      <quantity>2.000</quantity>
      <currency>AUD</currency>
      <amount>1.50</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.150000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142421144</usn>
      <lineNumber>1</lineNumber>
      <itemCode>001335</itemCode>
      <chargeFrom>2014-04-16+10:00</chargeFrom>
      <chargeTo>2014-05-01+10:00</chargeTo>
      <description>Retraction for Plan 1 (16/04/2014 - 30/04/2014)</
description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>-3.99</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>-0.500000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
```

```
      <usn>2142421144</usn>
      <lineNumber>2</lineNumber>
      <itemCode>001335</itemCode>
      <chargeFrom>2014-04-16+10:00</chargeFrom>
      <chargeTo>2014-05-01+10:00</chargeTo>
      <description>New charge</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>10.00</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>2.000000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>1000000008</usn>
      <lineNumber>3</lineNumber>
      <itemCode>round</itemCode>
      <chargeFrom>2014-04-15+10:00</chargeFrom>
      <chargeTo>2014-04-15+10:00</chargeTo>
      <description>Rounding adjustment</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>-0.01</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>round</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>1000000008</usn>
      <lineNumber>4</lineNumber>
      <itemCode>gst</itemCode>
      <chargeFrom>2014-04-15+10:00</chargeFrom>
      <chargeTo>2014-04-15+10:00</chargeTo>
      <description>Includes 10% GST</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>0.15</amount>
      <isGst>true</isGst>
      <chargeGst>false</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>gst</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142421144</usn>
      <lineNumber>5</lineNumber>
      <itemCode>gst</itemCode>
      <chargeFrom>2014-04-15+10:00</chargeFrom>
      <chargeTo>2014-04-15+10:00</chargeTo>
      <description>Includes 10% GST</description>
```

```
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>1.50</amount>
      <isGst>true</isGst>
      <chargeGst>false</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>gst</salesReportCategory>
    </transactionItem>
  </Invoice>
</PlanChangeResponse>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or another object referred to in the request does not exist.

`InvalidRequestException`

This fault is returned when the request is invalid.

`OfferExpiredException`

This fault is returned when changes have been made to any of the affected plans, or the subscription since the quote was produced. A new quote is required from `proposePlanChange()` before proceeding.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CommitPlanChange usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription of the plan to be changed.

***filename***

**Purpose:** Defines the filename of the PlanChangeOffer XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage pool reservations

Managing pool reservations lets you add and remove pool reservations, and retrieve information about existing pool reservations.

The following methods manage pool reservations:

- `getPoolReservationLinks()` returns all active pool reservation links of a subscription.
- `addIpRangeReservation()` adds an IP Range reservation to a subscription.
- `removePoolReservationLink()` removes a reservation link applied to a subscription.

### `getPoolReservationLinks()`

```
PoolReservationLinks getPoolReservationLinks(string usn)
```

This method retrieves all active pool reservation links of a subscription.

Any expired or deleted links will not be returned.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to retrieve the reservation links.

#### Results

This method returns a PoolReservationLinks XML document containing a list of active pool reservation links. The document lists the pool reservation link ID, the link type, pool reservation information and the expiry, if applicable.

There are no messages defined by this method.

#### Example: A returned PoolReservationLinks XML document

This example shows a returned PoolReservationLinks XML document specifying all pool reservation links.

```
<PoolReservationLinks xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <Link>
    <LinkId>d8ef89fa-f53d-4b8e-8775-3a7227da063b</LinkId>
    <LinkType>Quarantine</LinkType>
    <Reservation>
      <ReservationId>0cc6b0e8-3c20-4606-8085-69c94170f0c3</
ReservationId>
      <Start>10000100</Start>
      <End>10000199</End>
    </Reservation>
    <Expiry>2014-05-30T18:01:52.332+10:00</Expiry>
  </Link>
</PoolReservationLinks>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPoolReservationLinks usn
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to retrieve reservation links.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `addIpRangeReservation()`

```
SubscriptionIpRange addIpRangeReservation(string usn, string
 reservationId)
```

This method is used to add an IP range which was reserved through the Number Pools system to a given subscription. No IP range information is necessary as the information will be retrieved from the Number Pools system.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to which to add the reservation.

**reservationId**

**Format:** string

**Purpose:** Specifies a reservation ID from the reservation system. The reservation will be used to add an allocation link for the IP range to the subscription.

### Example: Request an IP range reservation

This example shows a request for an IP range reservation for subscription 1000120231.

```
usn: 1000120231
reservationId: 4CDD92B5-4F6B-4F17-B8AD-FA0706FB91E7
```

### Results

This method returns a SubscriptionIpRange XML document containing information about the IP range reservation. After the method returns successfully, the subscription will have created a link to the reserved IP range and will have updated its set of IP ranges using the reserved IP range's details.

Smile emits the SubscriptionIpRangesUpdated message when the task is complete.

### Example: A returned SubscriptionIpRange XML document

This example shows a SubscriptionIpRange XML document containing details of the IP range reservation for the given reservation ID.

```
<SubscriptionIpRange>
  <SubscritpionIpRangeId>C3AFD146-8ECB-4250-A770-F8ABB4C45D8D</
SubscritpionIpRangeId>
  <FromAddress>203.132.4.0</FromAddress>
  <ToAddress>203.132.4.255</ToAddress>
  <PoolReservationLink>
    <LinkId>6F91A091-8DCC-4D77-B810-DFB95DFED649</LinkId>
    <LinkType>Allocation</LinkType>
    <Reservation>
      <ReservationId>4CDD92B5-4F6B-4F17-B8AD-FA0706FB91E7</
ReservationId>
      <Start>203.132.4.0</Start>
      <End>203.132.4.255</End>
      <PoolType>IP</PoolType>
    </Reservation>
  </PoolReservationLink>
</SubscriptionIpRange>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or reservation is not found.

`InvalidRequestException`

This fault is returned when the request is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddIpRangeReservation usn reservationId
```

***usn***

**Purpose:** Specifies the USN of the subscription to which to add the reservation.

***reservationId***

**Purpose:** Specifies a reservation ID from the reservation system.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `removePoolReservationLink()`

```
PoolReservationLink removePoolReservationLink(string usn, string
 reservationLinkId)
```

This method removes a reservation link applied to a subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription from which to remove the reservation link.

**reservationLinkId**

**Format:** string

**Purpose:** Specifies the value representing the pool reservation link to remove from the subscription.

#### Results

This method returns a PoolReservationLink XML document containing a Quarantine link type if a reservation is removed. Otherwise an empty result is returned.

Smile emits the `SubscriptionIpRangesUpdated` message after the subscription is updated.

#### Example: A returned PoolReservationLink XML document

This example shows a returned PoolReservationLink XML document specifying that the pool reservation is quarantined.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>d8ef89fa-f53d-4b8e-8775-3a7227da063b</LinkId>
  <LinkType>Quarantine</LinkType>
  <Reservation>
    <ReservationId>0cc6b0e8-3c20-4606-8085-69c94170f0c3</
ReservationId>
    <Start>10000100</Start>
    <End>10000199</End>
  </Reservation>
  <Expiry>2014-05-30T18:01:52.332+10:00</Expiry>
</PoolReservationLink>
```

#### Faults

`NoSuchItemException`

This fault is returned when the subscription or reservationLinkId does not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RemovePoolReservationLink usn
```

```
reservationLinkId
```

***usn***

**Purpose:** Specifies the USN of the subscription from which to remove the reservation link.

***reservationLinkId***

**Purpose:** Specifies the value representing the pool reservation link to remove from the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage IP ranges

Manging IP ranges lets you retrieve, add and remove IP range reservations on subscriptions in Smile.

The following methods manage IP ranges:

- `getIpRanges()` retrieves all IP range reservations for a subscription.
- `addIpRange()` allocates a non-managed IP range reservation to a subscription.
- `removeIpRanges()` removes an IP range reservation from a subscription.

### `getIpRanges()`

```
SubscriptionIpRanges getIpRanges(string usn)
```

This method retrieves all IP addresses reserved from IP number pools, as well as those not managed by a pool reservation, for a given subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to retrieve IP range reservations.

#### Results

This method returns a SubscriptionIpRanges XML document containing a list of the IP address reservations for the given subscription. If the subscription has no reservations, an empty list will be returned.

There are no messages defined by this method.

#### Example: A returned SubscriptionIpRanges XML document

This example shows a SubscriptionIpRanges XML document.

```
<SubscriptionIpRanges xmlns="http://xml.inomial.com/smile/2.xsd">
  <SubscriptionIpRange>
    <SubscriptionIpRangeId>9E291E0C-0014-4205-8568-C6AC5C979234</
SubscriptionIpRangeId>
    <FromAddress>10.16.126.1</FromAddress>
    <ToAddress>10.16.126.254</ToAddress>
  </SubscriptionIpRange>
  <SubscriptionIpRange>
    <SubscriptionIpRangeId>1F9C9511-487C-4DEC-B0B9-3759ED80B0A7</
SubscriptionIpRangeId>
    <FromAddress>2001:0269:77c3:15ef::1</FromAddress>
    <ToAddress>2001:0269:77c3:153f:ffff:ffff:ffff:fffe</ToAddress>
  </SubscriptionIpRange>
</SubscriptionIpRanges>
```

### Example: A returned SubscriptionIpRanges XML document containing a range reserved from IP number pools

This example shows a SubscriptionIpRanges XML document containing a range reserved from IP number pools.

```
<SubscriptionIpRanges xmlns="http://xml.inomial.com/smile/2.xsd">
  <SubscriptionIpRange>
    <SubscriptionIpRangeId>61eda8bb-e573-4926-b094-899f30c3cc9a</
SubscriptionIpRangeId>
    <FromAddress>127.0.0.1</FromAddress>
    <ToAddress>127.0.0.1</ToAddress>
    <PoolReservationLink>
      <LinkId>9ada796d-51aa-43f8-b123-5ed843bd2fc4</LinkId>
      <LinkType>Allocation</LinkType>
      <Reservation>
        <ReservationId>3b02ab91-813f-421a-9b31-0341285645b3</
ReservationId>
        <Start>127.0.0.1</Start>
        <End>127.0.0.1</End>
      </Reservation>
    </PoolReservationLink>
  </SubscriptionIpRange>
</SubscriptionIpRanges>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetIpRanges usn
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to retrieve IP range reservations.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### addIpRange()

```
SubscriptionIpRange addIpRange(string usn, IpRange ipRange)
```

This method allocates a non-managed IP range reservation to a subscription. Note that other subscriptions can be allocated the same IP address and Smile will not generate a fault message.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to which the IP range reservation is added.

**ipRange**

**Format:** IpRange XML

**Purpose:** Defines the IP address range to be allocated to the subscription.

#### Example: An IpRange XML document

This example shows an IpRange XML document.

The following elements need to be provided:

| | |
|---|---|
| **`<FromAddress>`** | Textual representation of the lowest-numbered address of the IP range. The `FromAddress` parameter must be numerically lower than the `ToAddress`. |
| **`<ToAddress>`** | Textual representation of the highest-numbered address of the IP range. |

Both address strings must be in one of the following formats:

- A dotted-decimal numeric IPv4 address, as per RFC 790.
- A colon-interspersed hexadecimal IPv6 address, as per RFC 2373.

**Note:** Both elements must belong to the same IP address protocol; it is not valid to use the embedded-IPv4 address feature of IPv6 as described in section 2.5.4 of RFC 2373 if the other argument is an IPv4 address.

No check is made to establish if the IP range is already partially or wholly allocated to the same or another subscription, or if the IP range covers reserved or otherwise unusable addresses, such as loopback, private, broadcast or multicast addresses.

```
<IpRange xmlns="http://xml.inomial.com/smile/2.xsd">
  <FromAddress>192.168.212.24</FromAddress>
  <ToAddress>192.168.212.27</ToAddress>
</IpRange>
```

### Results

This method returns a SubscriptionIpRange XML document confirming the new allocation. The return result includes a distinct `<SubscriptionIpRangeId>` UUID string that identifies the IP range reservation. This UUID is required if the `removeIpRange()` method is later used to deallocate this reservation.

Smile emits the `SubscriptionIpRangesUpdated` message after the subscription is updated.

### Example: A returned SubscriptionIpRange XML document

This example shows a SubscriptionIpRange XML document.

```
<SubscriptionIpRange xmlns="http://xml.inomial.com/smile/2.xsd">
  <SubscriptionIpRangeId>2B1DB0AE-242F-4E95-BF19-0FD61FF8485D</
SubscriptionIpRangeId>
  <FromAddress>192.168.212.24</FromAddress>
  <ToAddress>192.168.212.27</ToAddress>
</SubscriptionIpRange>
```

### Faults

`InvalidRequestException`

This fault is returned when the FromAddress or ToAddress elements do not comply with IP address specifications or are malformed.

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddIpRange usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to which the IP range reservation is added.

***filename***

**Purpose:** Defines the filename of the IpRange XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `removeIpRange()`

```
void removeIpRange(string usn, string ipRangeId)
```

This method removes an IP range reservation from a subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription from which to delete the IP range reservation.

**ipRangeId**

**Format:** string

**Purpose:** Specifies the UUID of the IP range reservation to delete.

**Note:** The `ipRangeId` for a given IP range reservation is normally returned by the `addIpRange()` method. It can also be obtained via the `getIpRanges()` method.

#### Results

This method does not return any output for a successful removal.

Smile emits the `SubscriptionIpRangesUpdated` message after the subscription is updated.

#### Faults

`IllegalArgumentException`

This fault is returned when the ipRangeId parameter is malformed.

`NoSuchItemException`

This fault is returned when:

- the subscription given does not exist
- the IP range reservation does not exist
- the IP range reservation does not belong to the subscription given

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RemoveIpRange usn ipRangeId
```

***usn***

**Purpose:** Specifies the USN of the subscription from which to delete the IP range reservation.

***ipRangeId***

**Purpose:** Specifies the UUID of the IP range reservation to delete.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Download unrated activity statements

Activity statements, or CDRs, provide an itemised list of subscription usage over a period of time. For example, an unrated activity statement for a telephone subscription contains details of the calls made from the phone, including the call destinations and durations. Because the activity statement is unrated, it does not contain any information about call charges.

Downloading unrated activity statements helps you monitor network use and lets you use the statements in your billing processes. You can download unrated activity statements for an account or subscription at any time after they are created.

The following methods download unrated activity statements for an account or subscription:

- `getActivityStatementRange()` downloads unrated activity statements in a given date range. Use this method to display usage data to customers.
- `getActivityStatementBatch()` downloads unrated activity statements in batches. Use this method to retrieve activity statements for billing.

**Note:** To download the rated activity statements associated with an invoice, use `getRatings()` on the account endpoint.

### `getActivityStatementRange()`

```
ActivityBatchURL getActivityStatementRange(string usn,
 dateTime startTimestamp, dateTime endTimestamp)
```

This method returns a URL from which you can download unrated activity statements, or CDRs, that occur in a given date range.

You can only download information about activity that starts within the date range. For example, if a phone call begins at 10:59 am on 9 August 2012 and the date range begins at 11:00 am on 9 August 2012, the activity statement for the phone call is not included in the date range.

Use this method to download unrated activity statements for non-billing purposes, for example, to display activity statements to customers in a web interface.

**Warning:** To ensure that you do not exclude activity statements that are not bound to a subscription in timestamp order, do not use this method to download activity statements for billing. Instead, use the `getActivityStatementBatch()` method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription or account. If the USN is an account, Smile makes available the activity statements for all subscriptions associated with the account.

**startTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time that starts the date range, for example, 2012-05-01T00:00:00+11:00.

**endTimestamp**

**Format:** dateTime

**Purpose:** Specifies the date and time that ends the date range, for example, 2012-05-31T23:59:59+11:00.

### Results

This method returns a URL from which you can download the unrated activity statements. The URL is valid for 24 hours after it is generated by Smile and the activity statements are collated into a CSV file.

If there is no activity in the date range, the method returns activity="false".

This method and the URL both use basic access authentication. You must use the same credentials to access the URL as you did to call the method.

There are no messages defined by this method.

**Note:** For more information about the format of activity statements, see the CDR documentation under Format Guides in the Inomial Documentation Library.

### Example: A returned ActivityBatchURL XML document with activity statements

This example shows a returned ActivityBatchURL document with the URL from which you can download the activity statements.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="true">
  <BatchID>1409213</BatchID>
  <URL>https://smile.example.com/activity?token=12049IASUBF9812p9</
URL>
</ActivityBatchURL>
```

### Example: A returned ActivityBatchURL XML document without activity

This example shows a returned ActivityBatchURL document with no activity.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="false"/>
```

### Faults

`ActivityException`

This fault is returned when Smile cannot produce an activity statement.

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SubscriptionActivityRange usn
 startTimestamp endTimestamp
```

***usn***

**Purpose:** Specifies the USN of the account or subscription.

***startTimestamp***

**Purpose:** Specifies the date and time that starts the date range.

***endTimestamp***

**Purpose:** Specifies the date and time that ends the date range.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Download activity statements in batches

Downloading activity statements in batches ensures that you download all activity statements associated with a subscription, even if the activity statements are not bound to the subscription in timestamp order. For example, if an upstream provider sends you activity statements in batches, the activity statement may not be bound to the subscription until some time after the activity occurred.

The following process explains how to download activity statements in batches:

1. An application calls `getActivityStatementBatch()` and provides a batch ID of 0.
2. Smile creates a new batch against the USN, and returns a URL to download all activity bound to the subscription.
3. The application downloads the activity statements from the URL and persists the last batch ID in the download.
4. After a period of time, the application calls `getActivityStatementBatch()` and provides the persisted last batch ID.
5. Smile returns a URL to download all activity statements with a batch ID greater than the provided value.
6. The application downloads the activity statements from the URL, and persists the new last batch ID in the download.

The following diagram gives you an overview of how to download activity statements in batches.

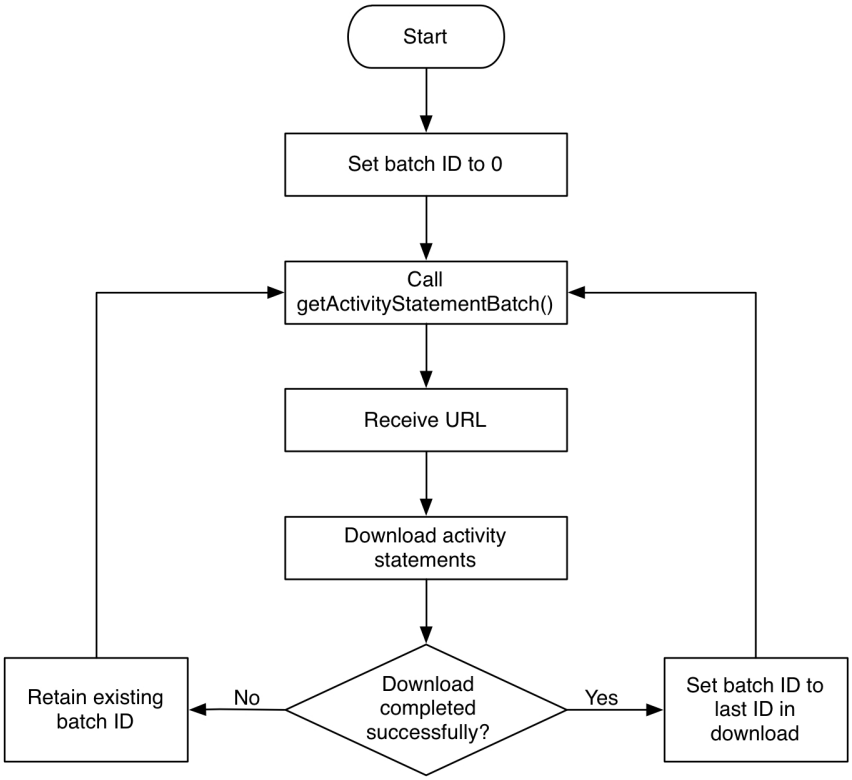



**Figure 8: Download activity statements in batches**

### getActivityStatementBatch()

```
ActivityBatchURL getActivityStatementBatch(string usn, long batchId)
```

This method returns a URL from which you can download a batch of unrated activity statements, or CDRs, for an account or subscription. Downloading activity statements in batches ensures that you download all activity statements for the USN, even if the activity is not recorded in timestamp order.

A caller can create multiple batches against a USN, and multiple callers can create batches against the same USN.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription or account. If the USN is an account, Smile makes available the activity statements for all subscriptions associated with the account.

**batchId**

**Format:** long datatype

**Purpose:** Specifies the batch ID of the last successfully downloaded batch; Smile makes available all activity statements with an ID after this batch ID. If you specify 0, Smile creates a new batch and makes available all the activity statements for the USN.

#### Results

This method returns a URL from which you can download the unrated activity statements. The URL is valid for 24 hours after it is generated by Smile and the activity statements are collated into a CSV file.

If there is no new activity to download, the method returns activity="false".

This method and the URL both use basic access authentication. You must use the same credentials to access the URL as you did to call the method.

There are no messages defined by this method.

**Note:** For more information about the format of activity statements, see the CDR documentation under Format Guides in the Inomial Documentation Library.

#### Example: A returned ActivityBatchURL XML document with activity statements

This example shows a returned ActivityBatchURL document with the URL from which you can download the activity statements.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="true">
  <BatchID>1409213</BatchID>
  <URL>https://smile.example.com/activity?token=12049IASUBF9812p9</
URL>
</ActivityBatchURL>
```

### Example: A returned ActivityBatchURL XML document without activity

This example shows a returned ActivityBatchURL document with no activity.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="false"/>
```

### Faults

`ActivityException`

This fault is returned when Smile cannot produce an activity statement.

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription, or the specified batch ID does not match an activity batch.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SubscriptionActivity usn batchID
```

***usn***

**Purpose:** Specifies the USN of the subscription or account.

***batchID***

**Purpose:** Specifies the batch ID of the last successfully downloaded batch.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Download rated activity statements

Activity statements, or CDRs, provide an itemised list of subscription usage over a period of time. For example, a rated activity statement for an ADSL subscription contains details of data downloads, data uploads and online time, and the charges associated with each billing item.

Downloading rated activity statements helps you track or estimate how much your customers spend in a given rating period. You can download rated activity statements after the pre-rating or rating process is complete.

The pre-rating process runs at intervals in open periods, and provides an indication of the value of subscription usage in the period to date. The rating engine runs at the end of the period; ratings produced by the rating engine do not change unless the period is reopened.

The following methods lets you download rated activity statements for a subscription:

1. `getRatingPeriods()` returns a list of rating periods for a subscription.
2. `getRatingsForPeriod()` downloads rated activity statements for a given rating period.

**Note:** To download the rated activity statements associated with an invoice, use `getRatings()` on the account endpoint.

### Rating and invoicing cycles

Rating and invoicing cycles have similar functions:

- **rating cycle**—specifies the length of time for which usage charges are calculated.
- **invoicing cycle**—specifies when a customer is sent an invoice.

Each instance of a rating or invoicing cycle is named a period. For example, a subscription may have a rating cycle that ends on the 14th of each month and a rating period of 15 May 2012-14 June 2012.

Each subscription has both a rating and invoicing cycle. The rating and invoicing cycles may end on the same date or have different end dates. Rating and invoicing cycles can end at the end of the month or on the subscription's anniversary. The anniversary is the day of the month on which the subscription was created.

If an account has multiple subscriptions, by default Smile sets all subscriptions to use the same invoicing cycle as the account. This lets you issue a single bill for the account each month.

When a billing run starts, Smile creates an invoice for each account that has a subscription with an invoicing period that ends on the date of the billing run. If an account contains two or more subscriptions with the same invoicing period end date, Smile adds the charges for both subscriptions to the same invoice.

#### Example: Rating and invoicing cycles

The following diagram represents a subscription with different end dates for the rating and invoicing cycles:

- The rating cycle begins on the 10th of each month and ends on the 9th of the following month.

- The invoicing cycle begins on the 15th of each month and ends on the 14th of the following month.

For example, on the invoicing period end date of 14 July, Smile will generate an invoice for the rating period of 10 June-9 July.

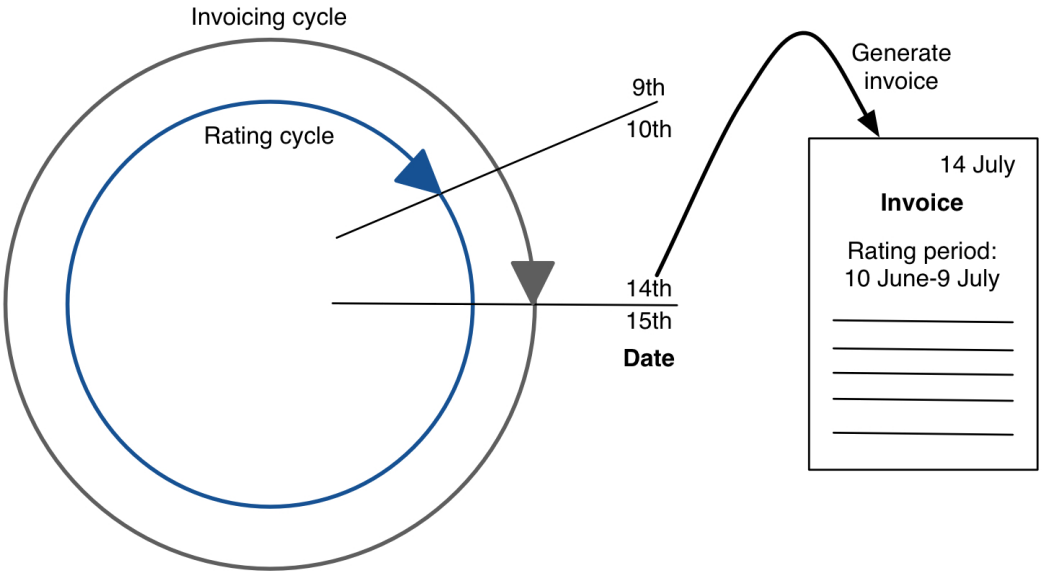



**Figure 9: Rating and invoicing cycles**

### `getRatingPeriods()`

```
RatingPeriodList getRatingPeriods(string usn)
```

This method returns information about a subscription's rating periods. Use this method to retrieve the rating period ID before you call `getRatingsForPeriod()`.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to retrieve rating period information.

#### Results

This method returns a RatingPeriodList XML document that contains information about the rating periods for which the subscription has usage data.

If the subscription is not yet activated, this method returns an empty list. If the subscription is closed, the status of each rating period is Closed.

There are no messages defined by this method.

### Example: A returned RatingPeriodList XML document

This example shows a returned RatingPeriodList XML document for a subscription with USN 2142420815. The `<releaseDate>` element specifies the date that the rating period becomes available for billing. Each `<ratingPeriodId>` is unique to this subscription.

The `<status>` element has one of the following values:

| | |
|---|---|
| **Open** | The billing run for the period has not begun, and usage can be associated with the rating period. |
| **Holding** | The invoicing period has been removed from automatic billing. Smile cannot set this status, but it may occur as a result of manual intervention. |
| **Waiting** | The billing run is processing and awaiting results. From this status, a rating period moves to Approving or Closed status. |
| **Approving** | The billing run is complete and awaiting approval. |
| **Closed** | The billing run is complete and approved. Any charges have been added to the subscription's account as an invoice or a set of deferred charges. |

This example shows two rating periods with Closed status, and three rating periods with Open status.

```
<RatingPeriodList>
  <RatingPeriodSummary>
    <USN>2142420815</USN>
    <ratingPeriodId>3045</ratingPeriodId>
    <invoicingPeriodEnd>2012-04-30+10:00</invoicingPeriodEnd>
    <ratingPeriodEnd>2012-04-30+10:00</ratingPeriodEnd>
    <releaseDate>2012-05-01+10:00</releaseDate>
    <status>Closed</status>
  </RatingPeriodSummary>
  <RatingPeriodSummary>
    <USN>2142420815</USN>
    <ratingPeriodId>3046</ratingPeriodId>
    <invoicingPeriodEnd>2012-05-31+10:00</invoicingPeriodEnd>
    <ratingPeriodEnd>2012-05-31+10:00</ratingPeriodEnd>
    <releaseDate>2012-06-01+10:00</releaseDate>
    <status>Closed</status>
  </RatingPeriodSummary>
  <RatingPeriodSummary>
    <USN>2142420815</USN>
    <ratingPeriodId>3047</ratingPeriodId>
    <invoicingPeriodEnd>2012-06-30+10:00</invoicingPeriodEnd>
    <ratingPeriodEnd>2012-06-30+10:00</ratingPeriodEnd>
    <releaseDate>2012-07-01+10:00</releaseDate>
    <status>Open</status>
  </RatingPeriodSummary>
  <RatingPeriodSummary>
    <USN>2142420815</USN>
    <ratingPeriodId>3048</ratingPeriodId>
    <invoicingPeriodEnd>2012-07-31+10:00</invoicingPeriodEnd>
    <ratingPeriodEnd>2012-07-31+10:00</ratingPeriodEnd>
```

```
    <releaseDate>2012-08-01+10:00</releaseDate>
    <status>Open</status>
  </RatingPeriodSummary>
  <RatingPeriodSummary>
    <USN>2142420815</USN>
    <ratingPeriodId>3049</ratingPeriodId>
    <invoicingPeriodEnd>2012-08-31+10:00</invoicingPeriodEnd>
    <ratingPeriodEnd>2012-08-31+10:00</ratingPeriodEnd>
    <releaseDate>2012-09-01+10:00</releaseDate>
    <status>Open</status>
  </RatingPeriodSummary>
</RatingPeriodList>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN is not a subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetRatingPeriods usn
```

***usn***

**Purpose:** Specifies the USN to retrieve rating periods for.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getRatingsForPeriod()`

```
ActivityBatchURL getRatingsForPeriod(int ratingPeriodId)
```

This method returns a URL from which you can download the rated activity statements, or CDRs, for a subscription by rating period. Downloading rated activity statements helps you track or estimate how much your customers spend in a given rating period.

You can download activity statements only after the rating or pre-rating process runs. You cannot use this method to download unrated activity statements.

This method returns all rated activity statements in the rating period. If you call this method twice in a rating period, Smile returns the statements at the start of the period twice.

Before you call this method, call `getRatingPeriods()` to retrieve the rating period ID. Each rating period ID is unique and identifies a rating period for a specific subscription.

### Parameters

**ratingPeriodId**

**Format:** integer

**Purpose:** Specifies the ID of the rating period to download activity statements for.

### Results

This method returns a URL from which you can download the rated activity statement. The URL is valid for 24 hours after it is generated by Smile. Activity statements are collated into a CSV file.

If there is no activity in the rating period, the method returns activity="false".

This method and the URL both use basic access authentication. You must use the same credentials to access the URL as you did to call the method.

There are no messages defined by this method.

**Note:** For more information about the format of activity statements, see the CDR documentation under Format Guides in the Inomial Documentation Library.

### Example: A returned ActivityBatchURL XML document with activity statements

This example shows a returned ActivityBatchURL document with the URL from which you can download the activity statements.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="true">
  <BatchID>1409213</BatchID>
  <URL>https://smile.example.com/activity?token=12049IASUBF9812p9</
URL>
</ActivityBatchURL>
```

### Example: A returned ActivityBatchURL XML document without activity

This example shows a returned ActivityBatchURL document with no activity.

```
<ActivityBatchURL xmlns="http://xml.inomial.com/smile/2.xsd"
 activity="false"/>
```

### Faults

`ActivityException`

This fault is returned when Smile cannot produce an activity statement.

`NoSuchItemException`

This fault is returned when the rating period ID does not match a rating period.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPeriodRatings ratingPeriodId
```

***ratingPeriodId***

**Purpose:** Specifies the rating period to return an activity statement for.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Synchronise subscription information

When a subscription changes, Smile sends a SmileSubscription message to the message bus to notify other applications of the change. Many actions can trigger a subscription change, for example, an action in Smile, a SOAP method or a customer exceeding their credit limit.

Listening for subscription messages:

- lets you synchronise subscription changes across your network
- lets other applications take action as a result of the subscription change

For example, you can configure an application to pass a provisioning request to an upstream provider after it receives a provisioning message, or configure a listener to gather subscription information for reports.

Smile emits SmileSubscription messages when:

- `CreditLimitExceeded`—a subscription's usage causes the account to exceed the subscription credit limit.
- `CreditLimitOkay`—an account that had exceeded its credit limit is now within the limit.
- `SubscriptionActivated`—billing is activated for a new subscription.
- `SubscriptionClosed`—a subscription is closed and billing is disabled.
- `SubscriptionEnabled`—a new subscription is authorised to access the network.
- `SubscriptionFeaturesChanged`—a subscription's features have changed.
- `SubscriptionPlanChanged`—a new plan schedule item on the subscription's accounting plan schedule becomes active.
- `SubscriptionProvisionCommand`—a command is invoked to provision a new feature for an existing subscription. This message is not emitted by default.
- `SubscriptionProvisioned`—provisioning is complete for a subscription.
- `SubscriptionProvisionResponse`—an authoritative source submits an internal message to affect the provisioning status of a subscription.
- `SubscriptionReadyToProvision`—a new subscription is ready to provision. This message is not emitted by default.
- `SubscriptionResumed`—a subscription is unsuspended, billing is re-activated and the subscription can re-access a service.
- `SubscriptionSpendLimitExceeded`—a subscription exceeds its spend limit.
- `SubscriptionSpendLimitReceded`—a subscription that had exceeded its spend limit is within the limit again.
- `SubscriptionSpendLimitStatusUpdated`—a subscription crosses a spend limit alert threshold, or a new rating period begins and spend limits for the subscription are reset.
- `SubscriptionSuspended`—a subscription's access to a service is temporarily suspended.
- `SubscriptionUpdated`—subscription data is changed.

### Related concepts

Listen for spend limit changes on page 280

### Related reference

SmileSubscription on page 489

Chapter

# 6

# Managing subservices

## Subservice API overview

A subservice is an identifier for a subscription. Often subservices represent multiple or alternative usernames of a subscription, for example a set of phone numbers for an ISDN telephony subscription.

The subservice WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/subservice?wsdl`

Related reference

Subservice endpoint on page 466

## Retrieve subservice information

Retrieving subservice information lets you view the current state of a subservice.

The following methods retrieve subservice information:

- getSubserviceBySubserviceUsername() returns a subservice that is linked to a given service, identified by unique service (SID) and subservice username.
- getSubservicesForSubscription() returns a list of subservices for a given subscription.
- getSubservice() returns a single subservice, identified by SSID.

### `getSubserviceBySubserviceUsername()`

```
getSubserviceBySubserviceUsername(int sid, string subserviceUsername)
```

This method retrieves a subservice that is linked to a particular service, identified by unique service (SID) and subservice username.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID that the requested subservice belongs to.

**subserviceUsername**

**Format:** string

**Purpose:** Specifies the username of the subservice for which information is required.

#### Results

This method returns a Subservice object with information of the requested subservice.

There are no messages defined by this method.

#### Example: A returned Subservice XML document

This example shows a Subservice XML document.

```
<Subservice xmlns="http://xml.inomial.com/smile/2.xsd">
  <SSID>2344</SSID>
  <SuscriptionUSN>9796932323</SubscriptionUSN>
  <Active>true</Active>
  <SubserviceUsername>joe@example.com</SubserviceUsername>
  <Object>
    <String name="type">emailAlias</String>
    <String name="forwardTo">john_smith@example.com</String>
  </Object>
</Subservice>
```

### Faults

`NoSuchItemException`

This fault is returned when the service ID (SID) provided does not correspond to an existing service or no subservices matching the given username exist for the specified service.

`TooManyResultsException`

This fault is returned when there is more than one subservice with the same given username. This can be the case if the service does not perform uniqueness checks on subservice usernames.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubserviceBySubserviceUsername
 sid subserviceUsername
```

***sid***

**Purpose:** Specifies the service ID.

***subserviceUsername***

**Purpose:** Specifies the username of the subservice.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSubservicesForSubscription()`

```
getSubservicesForSubscription(string usn)
```

This method returns a list of subservices for a specified subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to retrieve subservices for.

### Results

This method returns a SubserviceList containing zero or more Subservice objects corresponding to subservices for the given subscription.

There are no messages defined by this method.

### Example: A returned SubserviceList XML document

This example shows a SubserviceList XML document for subscription 9796932323.

```
<SubserviceList xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subservice>
    <SSID>2323</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>joe@example.com</SubserviceUsername>
```

```
    <Object>
      <String name="type">emailAlias</String>
      <String name="forwardTo">admin@example.com</String>
    </Object>
  </Subservice>
  <Subservice>
    <SSID>2324</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>mary@example.com</SubserviceUsername>
    <Object>
      <String name="type">emailAlias</String>
      <String name="forwardTo">admin@example.com</String>
    </Object>
  </Subservice>
  <Subservice>
    <SSID>2326</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>alice@example.com</SubserviceUsername>
    <Object>
      <String name="type">webHostAlias</String>
      <String name="redirectFrom">alice@example.com</String>
      <String name="redirectTo">admin@example.com</String>
    </Object>
  </Subservice>
</SubserviceList>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not correspond to an existing subscription in the database.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubservicesForSubscription usn
```

***usn***

**Purpose:** Specifies the USN of the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getSubservice()

```
getSubservice(string ssid)
```

This method returns a single subservice, identified by SSID.

#### Parameters

**ssid**

**Format:** string

**Purpose:** Specifies the subservice for which to retrieve information. This SSID is returned by addSubservice() or addSubservices().

#### Results

This method returns a single Subservice object describing the subservice requested.

There are no messages defined by this method.

#### Example: A returned Subservice object

This example shows a Subservice object for SSID 2344.

```
<Subservice xmlns="http://xml.inomial.com/smile/2.xsd">
  <SSID>2344</SSID>
  <SuscriptionUSN>9796932323</SubscriptionUSN>
  <Active>true</Active>
  <SubserviceUsername>joe@example.com</SubserviceUsername>
  <Object>
    <String name="type">emailAlias</String>
    <String name="forwardTo">joe_smith@example.com</String>
  </Object>
</Subservice>
```

#### Faults

`NoSuchItemException`

This fault is returned when the SSID does not correspond to an existing subservice.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubservice ssid
```

***ssid***

**Purpose:** Specifies the SSID of the subservice.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage subservices

Managing subservices lets you add, update and remove subservices.

The following methods manage subservices:

- addSubservice() creates a new subservice on an existing subscription.
- addSubservices() creates bulk subservices on the same subscription.
- removeSubservice() deletes a subservice.
- updateSubservice() changes information of a subservice.

### `addSubservice()`

```
Subservice addSubservice(string usn, SubserviceRequest subservice)
```

This method creates a new subservice associated with an existing subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription that the new subservice will be associated with.

**subservice**

**Format:** SubserviceRequest XML

**Purpose:** Defines a SubserviceRequest XML document that specifies the details of the subservice.

#### Example: A SubserviceRequest XML document

This example shows how to create a new subservice.

The following elements must be populated:

| | |
|---|---|
| **<SubscriptionUSN>** | Specifies the USN of the subscription. Must be same as the `usn` parameter. |
| **<Active>** | `true` or `false`<br>Specifies if the subservice is active. |
| **<SubserviceUsername>** | Specifies an alias name for the subservice. Any constraints on this name is determined by the service that the subscription is linked to. |
| **<Object>** | A custom object element that may be used to store additional information for the subservice. |

```
<SubserviceRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <SuscriptionUSN>9796932323</SubscriptionUSN>
  <Active>true</Active>
  <SubserviceUsername>joe@example.com</SubserviceUsername>
```

```
  <Object>
    <String name="type">emailAlias</String>
    <String name="forwardTo">joe@example.com</String>
  </Object>
</SubserviceRequest>
```

### Results

This method returns a Subservice XML document describing the new subservice. This document includes the SSID field that can be used by other subservice methods.

There are no messages defined by this method.

### Example: A returned Subservice XML document

This example shows a Subservice XML document for subscription 9796932323.

```
<Subservice xmlns="http://xml.inomial.com/smile/2.xsd">
  <SSID>2344</SSID>
  <SuscriptionUSN>9796932323</SubscriptionUSN>
  <Active>true</Active>
  <SubserviceUsername>joe@example.com</SubserviceUsername>
  <Object>
    <String name="type">emailAlias</String>
    <String name="forwardTo">joe@example.com</String>
  </Object>
</Subservice>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not correspond to an existing subscription.

`InvalidRequestException`

This fault is returned when either the proposed subservice username is malformed or another subservice or subscription with the same username already exists (according to the constraints set by the service).

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddSubservices usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription.

***filename***

**Purpose:** Defines the filename of the SubserviceRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### addSubservices()

```
addSubservices(string usn, SubserviceRequestList subserviceList)
```

This method creates bulk subservices in one method call. All subservices created will be associated with the same subscription. This method is the equivalent of invoking addSubservice() repeatedly for each subservice contained in the subserviceList parameter.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription that the new subservice will be associated with.

**subserviceList**

**Format:** SubserviceRequestList XML

**Purpose:** Defines a SubserviceRequestList XML document that specifies the details of the subservice.

#### Example: A SubserviceRequestList XML document

This example shows how to create three new subservices.

The following elements must be populated:

| | |
|---|---|
| **<SubscriptionUSN>** | Specifies the USN of the subscription. Must be same as the `usn` parameter. |
| **<Active>** | `true` or `false`<br>Specifies if the subservice is active. |
| **<SubserviceUsername>** | Specifies an alias name for the subservice. Any constraints on this name is determined by the service that the subscription is linked to. |
| **<Object>** | A custom object element that may be used to store additional information for the subservice. |

```
<SubserviceRequestList xmlns="http://xml.inomial.com/smile/2.xsd">
  <SubserviceRequest>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>joe@example.com</SubserviceUsername>
    <Object>
      <String name="type">emailAlias</String>
      <String name="forwardTo">admin@example.com</String>
    </Object>
  </SubserviceRequest>
  <SubserviceRequest>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>mary@example.com</SubserviceUsername>
    <Object>
```

```
      <String name="type">emailAlias</String>
      <String name="forwardTo">admin@example.com</String>
    </Object>
  </SubserviceRequest>
  <SubserviceRequest xmlns="http://xml.inomial.com/smile/2.xsd">
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>alice@example.com</SubserviceUsername>
    <Object>
      <String name="type">webHostAlias</String>
      <String name="redirectFrom">alica@example.com</String>
      <String name="redirectTo">admin@example.com</String>
    </Object>
  </SubserviceRequest>
</SubserviceRequestList>
```

### Results

This method returns a SubserviceList XML document that contains a list of Subscription objects that correspond to the subservices created. These are populated with unique SSIDs that can be used to identify each subservice in future.

There are no messages defined by this method.

### Example: A returned SubserviceList XML document

This example shows a SubserviceList XML document.

```
<SubserviceList xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subservice>
    <SSID>2323</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>larry@example.com</SubserviceUsername>
    <Object>
      <String name="type">emailAlias</String>
      <String name="forwardTo">stooges@example.com</String>
    </Object>
  </Subservice>
  <Subservice>
    <SSID>2324</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>curly@example.com</SubserviceUsername>
    <Object>
      <String name="type">emailAlias</String>
      <String name="forwardTo">stooges@example.com</String>
    </Object>
  </Subservice>
  <Subservice>
    <SSID>2326</SSID>
    <SuscriptionUSN>9796932323</SubscriptionUSN>
    <Active>true</Active>
    <SubserviceUsername>moe@example.com</SubserviceUsername>
    <Object>
```

```
      <String name="type">webHostAlias</String>
      <String name="redirectFrom">moe.example.com</String>
      <String name="redirectTo">stooges.example.com</String>
    </Object>
  </Subservice>
</SubserviceList>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not correspond to an existing subscription.

`InvalidRequestException`

This fault is returned when one of the proposed subservice username is malformed or another subservice or subscription with that same username already exists (according to the constraints set by the service).

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddSubservices usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription.

***filename***

**Purpose:** Defines the filename of the SubserviceRequestList XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `removeSubservice()`

```
void removeSubservice(string ssid)
```

This method deletes a subservice.

#### Parameters

**ssid**

**Format:** string

**Purpose:** Specifies the subservice to be removed. Use getSubservicesForSubscription() or getSubserviceBySubserviceUsername() to obtain the SSID of a service.

#### Results

No results are returned by this method.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when no subservice with the specified SSID exists.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RemoveSubservice ssid
```

***ssid***

**Purpose:** Specifies the subservice to be removed.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `updateSubservice()`

```
void updateSubservice(Subservice sNew)
```

This method alters information of a subservice.

#### Parameters

**sNew**

**Format:** Subservice XML

**Purpose:** Specifies a Subservice XML document that contains information about the subservice to be updated. The `<SSID>` and `<SubscriptionUSN>` elements must be set correctly for this purpose.

**Note:** It is not possible to change the SSID or SubscriptionUSN properties for a subservice once created. To re-assign a subservice to another subscription, you must delete the subservice via removeSubservice() and create another one via addSubservice().

### Example: A Subservice XML document

This example shows a Subservice XML document.

```
<Subservice xmlns="http://xml.inomial.com/smile/2.xsd">
  <SSID>2344</SSID>
  <SuscriptionUSN>9796932323</SubscriptionUSN>
  <Active>true</Active>
  <SubserviceUsername>joe@example.com</SubserviceUsername>
  <Object>
    <String name="type">emailAccount</String>
    <Integer name="quotaGB">10</Integer>
    <Boolean name="spamFilteringEnabled">true</Boolean>
  </Object>
</Subservice>
```

### Results

No results are returned by this method.

There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when:

- no subservice with the given SSID exists in the database
- no subscription with the given SubscriptionUSN exists in the database
- the subservice given by SSID does not belong to the subscription given by SubscriptionUSN

`InvalidRequestException`

This fault is returned when an attempt is made to rename a subservice to a username that is either:

- not correctly formed, as determined by the configuration of the service associated with the subscription
- already in use by another subservice or subscription, as determined by the configuration of the service associated with the subscription

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateSubservice filename
```

***filename***

**Purpose:** Specifies the filename of the Subservice XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Chapter

7

# Managing managed plans

## Managed plans API overview

Managed plans is an optional layer on top of regular plans. Managed plans are intended to be the primary plan management mechanism, including for Smile operators. All previously existing functionality is still available for use.

Actions you can perform on a managed plan include:

- enabling plan change fees
- managing CAT/SOAP plan changes
- configuring restrictions on what plans are available to a subscription

You can configure managed plans in Smile. However, you can use the managed plans endpoint of the SOAP API to manage the following:

- **managed plan group**—defines how plan changes behave, including charges, invoice line items and when changes occur. When a subscription enters a managed plan group it enables the plan changes defined in the group.
- **managed plan**—is a mapping from the managed plans system to the old plans system. Managed plan define the plan and plan options.
- **managed plan choice**—is an entry in a managed plan group defining the conditions of when a particular managed plan is available for member subscriptions.

The managed plan WDSL is located at `http://smile.example.com:port/stage/smile/ws/v2/subscription/plan?wsdl`

### Related reference

Managed plans endpoint on page 437

## Retrieve managed plan information

Retrieving managed plans information lets you view the state of managed plan groups and subscriptions.

The following methods retrieve managed plan information:

- `getCurrentManagedPlanGroup()` returns the details of the managed plan group that the subscription provided is in.
- `getManagedPlanGroup()` returns the details of the managed plan group specified.
- `getManagedPlanGroups()` returns a list of managed plan groups, and their choices, for the given service.
- `getManagedPlanChoices()` returns a list of possible managed plan changes.
- `getPlanSchedule()` returns the current and future managed plan changes.

### `getCurrentManagedPlanGroup()`

```
ManagedPlanGroup getCurrentManagedPlanGroup(string usn)
```

This method retrieves the details of the managed plan group that the specified subscription is in.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

#### Results

This method returns a ManagedPlanGroup XML document.

If the given subscription does not belong to a managed plan group, then null is returned

There are no messages defined by this method.

#### Example: A returned ManagedPlanGroup XML document

This example shows a ManagedPlanGroup XML document.

```
<ManagedPlanGroup xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanGroupId>321</ManagedPlanGroupId>
  <Name>test.managedPlan</Name>
  <SID>628</SID>
  <UpgradeTiming>CurrentPeriod</UpgradeTiming>
  <CrossgradeTiming>Immediately</CrossgradeTiming>
  <DowngradeTiming>NextPeriod</DowngradeTiming>
  <PrePaymentRequiredFor>None</PrePaymentRequiredFor>
  <ManagedPlanChoices>
    <ManagedPlan>
      <ManagedPlanChoiceId>321</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan1</DisplayName>
      <Description>test.managedPlan.managedPlan1</Description>
```

```
      <Position>1</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
    <ManagedPlan>
      <ManagedPlanChoiceId>322</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan2</DisplayName>
      <Description>test.managedPlan.managedPlan2</Description>
      <Position>2</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
  </ManagedPlanChoices>
</ManagedPlanGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCurrentManagedPlanGroup usn
```

***usn***

**Purpose:** Specifies the USN of the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getManagedPlanGroup()`

```
ManagedPlanGroup getManagedPlanGroup(string managedPlanGroupId)
```

This method retrieves the details of the specified managed plan group.

### Parameters

**managedPlanGroupId**

**Format:** string

**Purpose:** Specifies the ID of a managed plan group.

### Results

This method returns a ManagedPlanGroup XML document.

There are no messages defined by this method.

### Example: A returned ManagedPlanGroup XML document

This example shows a ManagedPlanGroup XML document.

```
<ManagedPlanGroup xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanGroupId>321</ManagedPlanGroupId>
  <Name>test.managedPlan</Name>
  <SID>628</SID>
  <UpgradeTiming>CurrentPeriod</UpgradeTiming>
  <CrossgradeTiming>Immediately</CrossgradeTiming>
  <DowngradeTiming>NextPeriod</DowngradeTiming>
  <PrePaymentRequiredFor>None</PrePaymentRequiredFor>
  <ManagedPlanChoices>
    <ManagedPlan>
      <ManagedPlanChoiceId>321</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan1</DisplayName>
      <Description>test.managedPlan.managedPlan1</Description>
      <Position>1</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
    <ManagedPlan>
      <ManagedPlanChoiceId>322</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan2</DisplayName>
      <Description>test.managedPlan.managedPlan2</Description>
      <Position>2</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
  </ManagedPlanChoices>
</ManagedPlanGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetManagedPlanGroup
 managedPlanGroupId
```

***managedPlanGroupId***

**Purpose:** Specifies the ID of a managed plan group.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getManagedPlanGroups()

```
ManagedPlanGroupList getManagedPlanGroups(int sid)
```

This method retrieves a list of managed plan groups, and their choices, for the specified service.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the SID of a service.

#### Results

This method returns a ManagedPlanGroupList XML document that contains the managed plan groups of the specified service.

There are no messages defined by this method.

#### Example: A returned ManagedPlanGroupList XML document

This example shows a ManagedPlanGroupList XML document.

```
<ManagedPlanGroupList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanGroup>
    <ManagedPlanGroupId>321</ManagedPlanGroupId>
    <Name>test.managedPlan</Name>
    <SID>628</SID>
    <UpgradeTiming>CurrentPeriod</UpgradeTiming>
    <CrossgradeTiming>Immediately</CrossgradeTiming>
    <DowngradeTiming>NextPeriod</DowngradeTiming>
    <PrePaymentRequiredFor>None</PrePaymentRequiredFor>
    <ManagedPlanChoices>
      <ManagedPlan>
        <ManagedPlanChoiceId>321</ManagedPlanChoiceId>
        <DisplayName>test.managedPlan.managedPlan1</DisplayName>
        <Description>test.managedPlan.managedPlan1</Description>
        <Position>1</Position>
        <ManagedPlanGroupRef>
          <ManagedPlanGroupId>321</ManagedPlanGroupId>
          <Name>test.managedPlan</Name>
        </ManagedPlanGroupRef>
        <ManagedPlanOptions/>
      </ManagedPlan>
      <ManagedPlan>
        <ManagedPlanChoiceId>322</ManagedPlanChoiceId>
        <DisplayName>test.managedPlan.managedPlan2</DisplayName>
        <Description>test.managedPlan.managedPlan2</Description>
        <Position>2</Position>
        <ManagedPlanGroupRef>
          <ManagedPlanGroupId>321</ManagedPlanGroupId>
          <Name>test.managedPlan</Name>
        </ManagedPlanGroupRef>
        <ManagedPlanOptions/>
      </ManagedPlan>
```

```
    </ManagedPlanChoices>
  </ManagedPlanGroup>
</ManagedPlanGroupList>
```

### Faults

`NoSuchItemException`

This fault is returned when the SID provided does not have a corresponding service.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetManagedPlanGroups sid
```

***sid***

**Purpose:** Specifies the SID of a service.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getManagedPlanChoices()`

```
ManagedPlanQuoteList getManagedPlanChoices(string usn)
```

This method retrieves a list of possible managed plan changes for the specified subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

### Results

This method returns a ManagedPlanQuoteList XML document that contains the possible managed plan choices for the specified subscription.

**Warning:** The quotes generated by this method are not stored. They are not visible in Smile and you cannot refer to them.

There are no messages defined by this method.

### Example: A returned ManagedPlanQuoteList XML document

This example shows a ManagedPlanQuoteList XML document.

```
<ManagedPlanQuoteList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanQuote>
    <ManagedPlan>
      <ManagedPlanChoiceId>321</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan1</DisplayName>
      <Description>test.managedPlan.managedPlan1</Description>
      <Position>1</Position>
      <ManagedPlanGroupRef>
```

```
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
    <EffectiveDate>2014-12-17+11:00</EffectiveDate>
    <Direction>Crossgrade</Direction>
    <Quote>
      <version>2.0</version>
      <transactionType>Quote</transactionType>
      <company>1</company>
      <usn>2142422415</usn>
      <transactionNumber>21429899</transactionNumber>
      <currency>AUD</currency>
      <amount>1.20</amount>
      <accountType>1</accountType>
      <gstAmount>0.11</gstAmount>
      <salesReportCategory>invoice</salesReportCategory>
      <formattedAmount currency="AUD" formatted="$1.20">1.20</
formattedAmount>
      <formattedGstAmount currency="AUD" formatted="$0.11">0.11</
formattedGstAmount>
      <unallocatedAmount>1.20</unallocatedAmount>
      <formattedUnallocatedAmount currency="AUD"
 formatted="$1.20">1.20</formattedUnallocatedAmount>
      <transactionDate>2014-12-17+11:00</transactionDate>
      <openDate>2014-12-17+11:00</openDate>
      <quoteNumber>21429899</quoteNumber>
      <createdByUsn>2142420781</createdByUsn>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>0</lineNumber>
        <itemCode>004692</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Invoice item description</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>1.10</amount>
        <isGst>false</isGst>
        <chargeGst>true</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.11</gstAmount>
        <salesReportCategory>unknown</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>1</lineNumber>
        <itemCode>round</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Rounding adjustment</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>-0.01</amount>
        <isGst>false</isGst>
```

```
        <chargeGst>true</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0</gstAmount>
        <salesReportCategory>round</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>2</lineNumber>
        <itemCode>gst</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Includes 10% GST</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>0.11</amount>
        <isGst>true</isGst>
        <chargeGst>false</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.00</gstAmount>
        <salesReportCategory>gst</salesReportCategory>
      </transactionItem>
    </Quote>
  </ManagedPlanQuote>
  <ManagedPlanQuote>
    <ManagedPlan>
      <ManagedPlanChoiceId>322</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan2</DisplayName>
      <Description>test.managedPlan.managedPlan2</Description>
      <Position>2</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
    <EffectiveDate>2014-12-01+11:00</EffectiveDate>
    <Direction>Upgrade</Direction>
    <Quote>
      <version>2.0</version>
      <transactionType>Quote</transactionType>
      <company>1</company>
      <usn>2142422415</usn>
      <transactionNumber>21429907</transactionNumber>
      <currency>AUD</currency>
      <amount>0.10</amount>
      <accountType>1</accountType>
      <gstAmount>0.01</gstAmount>
      <salesReportCategory>invoice</salesReportCategory>
      <formattedAmount currency="AUD" formatted="$0.10">0.10</
formattedAmount>
      <formattedGstAmount currency="AUD" formatted="$0.01">0.01</
formattedGstAmount>
      <unallocatedAmount>0.10</unallocatedAmount>
      <formattedUnallocatedAmount currency="AUD"
 formatted="$0.10">0.10</formattedUnallocatedAmount>
```

```
      <transactionDate>2014-12-17+11:00</transactionDate>
      <openDate>2014-12-17+11:00</openDate>
      <quoteNumber>21429907</quoteNumber>
      <createdByUsn>2142420781</createdByUsn>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>0</lineNumber>
        <itemCode>004691</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Invoice item description</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>0.10</amount>
        <isGst>false</isGst>
        <chargeGst>true</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.01</gstAmount>
        <salesReportCategory>unknown</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>1</lineNumber>
        <itemCode>round</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Rounding adjustment</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>-0.01</amount>
        <isGst>false</isGst>
        <chargeGst>true</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0</gstAmount>
        <salesReportCategory>round</salesReportCategory>
      </transactionItem>
      <transactionItem>
        <usn>2142422423</usn>
        <lineNumber>2</lineNumber>
        <itemCode>gst</itemCode>
        <chargeFrom>2014-12-17+11:00</chargeFrom>
        <chargeTo>2014-12-17+11:00</chargeTo>
        <description>Includes 10% GST</description>
        <quantity>1</quantity>
        <currency>AUD</currency>
        <amount>0.01</amount>
        <isGst>true</isGst>
        <chargeGst>false</chargeGst>
        <discount>0</discount>
        <count>0</count>
        <gstAmount>0.00</gstAmount>
        <salesReportCategory>gst</salesReportCategory>
      </transactionItem>
    </Quote>
  </ManagedPlanQuote>
```

```
</ManagedPlanQuoteList>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription.

`PlanChangeUnavailableException`

This fault is returned when the subscription is not eligible for plan changes. For example, if the subscription does not belong to a plan change group.

`PlanChangeException`

This fault is returned when an internal error or configuration problem prevents the request from being processed.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetManagedPlanChoices usn
```

***usn***

**Purpose:** Specifies the USN of the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getPlanSchedule()

```
ManagedPlanSchedule getPlanSchedule(string usn)
```

This method retrieves the current and future managed plan changes for the specifies subscription.

This method will fail if custom plans are scheduled in these periods .

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

#### Results

This method returns a ManagedPlanSchedule XML document with that contains current and future managed plans.

There are no messages defined by this method.

#### Example: A returned ManagedPlanSchedule XML document

This example shows a ManagedPlanSchedule XML document.

```
<ManagedPlanSchedule xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanScheduleItem>
    <Start>2014-12-01T00:00:00.000+11:00</Start>
    <ManagedPlan>
      <ManagedPlanChoiceId>321</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan1</DisplayName>
      <Description>test.managedPlan.managedPlan1</Description>
      <Position>1</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
  </ManagedPlanScheduleItem>
</ManagedPlanSchedule>
```

#### Example: A returned ManagedPlanSchedule XML document

This example shows a ManagedPlanSchedule XML document.

```
<ManagedPlanSchedule xmlns+"http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <ManagedPlanScheduleItem>
    <Start>2014-12-01T00:00:00.000+11:00</Start>
    <ManagedPlan>
      <ManagedPlanChoiceId>322</ManagedPlanChoiceId>
      <DisplayName>test.managedPlan.managedPlan2</DisplayName>
      <Description>test.managedPlan.managedPlan2</Description>
```

```
      <Position>2</Position>
      <ManagedPlanGroupRef>
        <ManagedPlanGroupId>321</ManagedPlanGroupId>
        <Name>test.managedPlan</Name>
      </ManagedPlanGroupRef>
      <ManagedPlanOptions/>
    </ManagedPlan>
  </ManagedPlanScheduleItem>
</ManagedPlanSchedule>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription.

`PlanChangeUnavailableException`

This fault is returned when the subscription is not eligible for plan changes. For example, if the subscription does not belong to a plan change group.

`PlanChangeException`

This fault is returned when an internal error or configuration problem prevents the request from being processed.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPlanSchedule usn
```

***usn***

**Purpose:** Specifies the USN of the subscription.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage managed plans

Managing managed plans lets you change a subscription's managed plan.

The following methods manage managed plans:

- `changePlan()` changes the plan of the provided subscription.
- `changeManagedPlanGroup()` changes the managed plan group that the subscription belongs to.

### `changePlan()`

```
ManagedPlanChangeResponse changePlan(string usn,
 string managedPlanChoiceId, PaymentRequest paymentRequest,
 PlanOptions planOptions)
```

This method changes the plan of the specified subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

**managedPlanChoiceId**

**Format:** string

**Purpose:** Specifies the ID of a managed plan choice.

**paymentRequest**

**Format:** PaymentRequest XML

**Purpose:** (Optional) Defines a PaymentRequest XML document that specifies the payment details for the plan change.

**Note:** Required if the plan change requires pre-payment before processing the change.

**planOptions**

**Format:** PlanOptions XML

**Purpose:** (Optional) Defines a PlanOptions XML document that specifies plan option overrides for the plan change.

The PlanOptions XML type is ObjectType. Plan options and sub-elements can only be boolean or integer, with name as the name of the option, and the value is the value of the option.

**Note:** This element is optional, but as a plan can specify that certain plan options are required, billing errors may be produced if required options are not correctly configured.

#### Results

This method returns a ManagedPlanChangeResponse XML document that details the plan change, for example, invoice and payment details.

Smile emits a `SubscriptionPlanChanged` message after the plan is changed. If the plan change results in the subscription's features to change Smile emits the `SubscriptionFeaturesChanged` message. If this method raises financial transactions, Smile emits the `TransactionClosed` message.

### Example: A returned ManagedPlanChangeResponse XML document

This example shows a ManagedPlanChangeResponse XML document.


```
<ManagedPlanChangeResponse xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PlanScheduleItem>
    <PlanCode>tp6282</PlanCode>
    <PlanScheduleType>1</PlanScheduleType>
    <Start>2014-12-01T00:00:00.000+11:00</Start>
    <Options>
      <Object/>
    </Options>
    <PlanScheduleItem>1079</PlanScheduleItem>
  </PlanScheduleItem>
  <Invoice>
    <version>2.0</version>
    <transactionType>Quote</transactionType>
    <company>1</company>
    <usn>2142422415</usn>
    <transactionNumber>21429915</transactionNumber>
    <currency>AUD</currency>
    <amount>0.10</amount>
    <accountType>1</accountType>
    <gstAmount>0.01</gstAmount>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$0.10">0.10</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.01">0.01</
formattedGstAmount>
    <unallocatedAmount>0.10</unallocatedAmount>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$0.10">0.10</formattedUnallocatedAmount>
    <transactionDate>2014-12-17+11:00</transactionDate>
    <openDate>2014-12-17+11:00</openDate>
    <quoteNumber>21429915</quoteNumber>
    <createdByUsn>2142420781</createdByUsn>
    <transactionItem>
      <usn>2142422423</usn>
      <lineNumber>0</lineNumber>
      <itemCode>004691</itemCode>
      <chargeFrom>2014-12-17+11:00</chargeFrom>
      <chargeTo>2014-12-17+11:00</chargeTo>
      <description>Invoice item description</description>
      <quantity>1</quantity>
      <currency>AUD</currency>
      <amount>0.10</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0</discount>
      <count>0</count>
      <gstAmount>0.01</gstAmount>
```

```
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142422423</usn>
      <lineNumber>1</lineNumber>
      <itemCode>round</itemCode>
      <chargeFrom>2014-12-17+11:00</chargeFrom>
      <chargeTo>2014-12-17+11:00</chargeTo>
      <description>Rounding adjustment</description>
      <quantity>1</quantity>
      <currency>AUD</currency>
      <amount>-0.01</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0</discount>
      <count>0</count>
      <gstAmount>0</gstAmount>
      <salesReportCategory>round</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142422423</usn>
      <lineNumber>2</lineNumber>
      <itemCode>gst</itemCode>
      <chargeFrom>2014-12-17+11:00</chargeFrom>
      <chargeTo>2014-12-17+11:00</chargeTo>
      <description>Includes 10% GST</description>
      <quantity>1</quantity>
      <currency>AUD</currency>
      <amount>0.01</amount>
      <isGst>true</isGst>
      <chargeGst>false</chargeGst>
      <discount>0</discount>
      <count>0</count>
      <gstAmount>0.00</gstAmount>
      <salesReportCategory>gst</salesReportCategory>
    </transactionItem>
  </Invoice>
</ManagedPlanChangeResponse>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription, or managed plan change choice ID does not have a corresponding managed plan change choice.

`PlanChangeUnavailableException`

This fault is returned when the subscription is not eligible for plan changes. For example, if the subscription does not belong to a plan change group.

`PlanChangeException`

This fault is returned when an internal error or configuration problem prevents the request from being processed.

`InvalidRequestException`

This fault is returned when the plan change request is invalid. For example, when the provided USN is an account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ChangePlan usn managedPlanChoiceId
 [paymenRequest] [filename]
```

***usn***

**Purpose:** Specifies the USN of the subscription.

***managedPlanChoiceId***

**Purpose:** Specifies the ID of the managed plan choice.

***paymentRequest***

**Purpose:** (Optional) Defines the filename of the PaymentRequest XML document.

***filename***

**Purpose:** (Optional) Defines the filename of the PlanOptions XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### changeManagedPlanGroup()

```
void changeManagedPlanGroup(string usn, string
 managedPlanGroupId, string managedPlanChoiceId,
 ManagedPlanChangeTiming managedPlanChangeTiming, PlanOptions
 planOptions)
```

This method changes the managed plan group that a subscription belongs to.

Note that this operation does not raise any charges as the change is between two groups.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

**managedPlanGroupId**

**Format:** string

**Purpose:** Specifies the ID of a managed plan group.

**managedPlanChoiceId**

**Format:** string

**Purpose:** Specifies the ID of a managed plan choice. The plan choice must correspond to the managed plan group.

**managedPlanChangeTiming**

**Format:** ManagedPlanChangeTiming XML

**Purpose:** Specifies the details of when the plan change is effective.

**planOptions**

**Format:** PlanOptions XML

**Purpose:** (Optional) Specifies plan option overrides for the plan change.

The PlanOptions XML type is ObjectType. Plan options and sub-elements can only be boolean or integer, with name as the name of the option, and the value is the value of the option.

**Note:** This element is optional, but as a plan can specify that certain plan options are required, billing errors may be produced if required options are not correctly configured.

#### Results

This method returns void.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not have a corresponding subscription.

`PlanChangeException`

This fault is returned when an internal error or configuration problem prevents the request from being processed.

`InvalidRequestException`

This fault is returned when the USN, managedPlanGroupId and managedPlanChoiceId are not compatible.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ChangeManagedPlanGroup usn
 managedPlanGroupId managedPlanChoiceId managedPlanChangeTiming
 [filename]
```

***usn***

**Purpose:** Specifies the USN of the subscription.

***mangedPlanGroupId***

**Purpose:** Specifies the ID of a managed plan group.

***managedPlanChoiceId***

**Purpose:** Specifies the ID of a managed plan choice.

***managedPlanChangeTiming***

**Purpose:** Defines the filename of the ManagedPlanChangeTiming XML document.

***filename***

**Purpose:** (Optional) Defines the filename of the PlanOptions XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

# Chapter 8

# Managing usage and spend

## Spend management API overview

The spend management endpoint of the SOAP API lets you retrieve information about:

- **value pools**—the monetary value of a subscription's included usage
- **usage alert groups**—the physical amount of a subscription's included usage, for example, included download limits
- **spend limits**—the amount that a subscription can spend on a particular usage type per rating period, for example, $100 on premium SMS services

The methods on this endpoint let you retrieve information about subscription usage and spend. You cannot use this endpoint to manage value pools, usage alert groups or spend limits for a service or plan.

You can perform the following tasks on the spend management endpoint:

- retrieve the state of a subscription's value pools, usage alert groups or spend limits
- set a temporary or permanent spend limit override
- change the alert thresholds for a subscription's spend limit

You must enable pre-rating before you use the spend management API. Pre-rating is a process that runs at intervals during an open period and provides an indication of subscription usage in the period to date. Contact Inomial for more information about enabling pre-rating.

The spend management WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/spendmanagement?wsdl`

### Related reference

Spend management endpoint on page 457

## Alert thresholds

An alert threshold is a percentage of value that, when consumed, triggers a notification and raises a message. Alert thresholds apply to value pools, usage alert groups and spend limits.

Alert thresholds help you manage customer spend and prevent bill shock. You can configure Smile to notify customers when they exceed an alert threshold and consume a certain percentage of their included value.

Default alert thresholds are defined on a service or plan, but you can customise the alert thresholds for a subscription. Active alert thresholds are named effective alert thresholds. If you wish, you can write an application that lets customers choose their own effective alert thresholds.

## Retrieve value pool states

A value pool tracks the monetary value of included usage that a subscription spends in a rating period. A subscription can have multiple value pools, for example, a mobile subscription may have a $100 value pool for local calls and a $50 value pool for mobile data downloads.

Smile resets each value pool to $0 at the start of each rating period and when a subscription changes plan. The value pool limit is pro-rated for incomplete rating periods.

Retrieving value pool states lets you view information about the amount of included value that a customer has spent in the current rating period.

The following methods retrieve value pool states:

- `getSubscriptionValuePoolStates()` returns information about all value pools for a subscription.
- `getSubscriptionValuePoolState()` returns information about a specified value pool.

### `getSubscriptionValuePoolStates()`

```
SubscriptionValuePoolState getSubscriptionValuePoolStates(string usn)
```

This method returns information about the current state of each value pool associated with a subscription. Use this method to retrieve and display information about a subscription's value pools.

**Note:** You must enable pre-rating before you use this method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the subscription associated with the value pools.

### Results

This method returns a SubscriptionValuePoolState XML document that lists information about the current state of each of the subscription's value pools, for example, the value pool limit, amount spent in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

### Example: A returned SubscriptionValuePoolState XML document

This example shows a SubscriptionValuePoolState XML document for a subscription with two value pools. The Included Local Calls and SMS value pool has a limit of $500 and the Included Calls value pool has a limit of $50. The balance of both value pools is $0.

```
<SubscriptionValuePoolState xmlns="http://xml.inomial.com/
smile/2.xsd">
  <SubscriptionValuePool>
    <ValuePool>
      <valuePoolId>252</valuePoolId>
      <sid>137</sid>
      <name>Included Local Calls and SMS</name>
      <alertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </alertThresholds>
    </ValuePool>
    <periodEnd>2012-08-31+12:00</periodEnd>
    <limit>500.00</limit>
    <effectiveAlertThresholds>
      <alertThreshold>50</alertThreshold>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </effectiveAlertThresholds>
    <currentSpend>0.00</currentSpend>
    <currentThreshold>0</currentThreshold>
    <previousThreshold>0</previousThreshold>
    <resetDate>2012-09-01T00:00:00.000+12:00</resetDate>
  </SubscriptionValuePool>
  <SubscriptionValuePool>
    <ValuePool>
      <valuePoolId>183</valuePoolId>
      <sid>137</sid>
      <name>Included Calls</name>
      <alertThresholds>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </alertThresholds>
    </ValuePool>
    <periodEnd>2012-08-31+12:00</periodEnd>
    <limit>50.00</limit>
    <effectiveAlertThresholds>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
```

```
    </effectiveAlertThresholds>
    <currentSpend>0.00</currentSpend>
    <currentThreshold>0</currentThreshold>
    <previousThreshold>0</previousThreshold>
    <resetDate>2012-09-01T00:00:00.000+12:00</resetDate>
  </SubscriptionValuePool>
</SubscriptionValuePoolState>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionValuePoolStates
 usn
```

***usn***

**Purpose:** Specifies the subscription associated with the value pools.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSubscriptionValuePoolState()`

```
SubscriptionValuePool getSubscriptionValuePoolState(string usn,
 string valuePoolId)
```

This method returns information about the current state of a specified value pool. Use this method to retrieve and display information about a specific value pool for a subscription.

**Note:** You must enable pre-rating before you use this method.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the subscription associated with the value pool.

**valuePoolId**

**Format:** string

**Purpose:** Specifies the ID of the value pool for which you want to retrieve information.

### Results

This method returns a SubscriptionValuePoolState XML document that lists information about the current state of the value pool, for example, the value pool limit, amount spent in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

### Example: A returned SubscriptionValuePool XML document

This example shows a SubscriptionValuePool XML document for the Included Calls value pool. The value pool has a limit of $50 and alert thresholds at 80% and 100% of the value pool. Smile resets the value pool to $0 at midnight on 1 September 2012.

```
<SubscriptionValuePool xmlns="http://ws.inomial.com/smile.2"
 xmlns="http://xml.inomial.com/smile/2.xsd">
  <ValuePool>
    <valuePoolId>183</valuePoolId>
    <sid>137</sid>
    <name>Included Calls</name>
    <alertThresholds>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </alertThresholds>
  </ValuePool>
  <periodEnd>2012-08-31+12:00</periodEnd>
  <limit>50.00</limit>
  <effectiveAlertThresholds>
    <alertThreshold>80</alertThreshold>
    <alertThreshold>100</alertThreshold>
   </effectiveAlertThresholds>
  <currentSpend>0.00</currentSpend>
  <currentThreshold>0</currentThreshold>
  <previousThreshold>0</previousThreshold>
  <resetDate>2012-09-01T00:00:00.000+12:00</resetDate>
</SubscriptionValuePool>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or value pool does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionValuePoolState
        usn valuePoolId
```

***usn***

**Purpose:** Specifies the subscription associated with the value pool.

***valuePoolId***

**Purpose:** Specifies the ID of the value pool for which you want to retrieve information.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Retrieve usage alert group states

A usage alert group tracks the physical amount of included usage that a subscription consumes in a rating period. A subscription can have multiple usage alert groups, for example, a mobile subscription may have a usage alert group of 100 minutes for local calls and a usage alert group of 1 GB for mobile data downloads.

Smile resets the value of each usage alert group to 0 at the start of each rating period and when a subscription changes plan. The usage alert group limit is pro-rated for incomplete rating periods. Smile can perform an action when a subscription exceeds the value of a usage alert group, for example, suspend or shape the subscription.

Retrieving usage alert group states lets you view information about the quantity of included usage that a customer has consumed in the current rating period.

The following methods retrieve usage alert group states:

- `getSubscriptionUsageAlertGroups()` returns information about all usage alert groups for a subscription.
- `getSubscriptionUsageAlertGroup()` returns information about a specified usage alert group.

### `getSubscriptionUsageAlertGroups()`

```
SubscriptionUsageAlertGroups getSubscriptionUsageAlertGroups(string
 usn)
```

This method returns information about the current state of each usage alert group associated with a subscription. Use this method to retrieve and display information about a subscription's usage alert groups.

**Note:** You must enable pre-rating before you use this method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the subscription associated with the usage alert groups.

#### Results

This method returns a SubscriptionUsageAlertGroups XML document that lists information about the current state of the subscription's usage alert groups, for example, the usage alert group limit, amount consumed in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

### Example: A returned SubscriptionUsageAlertGroups XML document

This example shows a SubscriptionUsageAlertGroups document for a subscription with a single usage alert group for mobile data usage. The subscription has exceeded the usage alert group limit for the current rating period.

```
<SubscriptionUsageAlertGroups xmlns="http://xml.inomial.com/
smile/2.xsd">
  <SubscriptionUsageAlertGroup>
    <UsageAlertGroup>
      <usageAlertGroupID>3</usageAlertGroupID>
      <company>1</company>
      <name>Mobile Data Usage</name>
      <alertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </alertThresholds>
    </UsageAlertGroup>
    <periodEnd>2012-08-31+09:30</periodEnd>
    <limit>1000000</limit>
    <currentUse>211929000</currentUse>
    <currentThreshold>100</currentThreshold>
  </SubscriptionUsageAlertGroup>
</SubscriptionUsageAlertGroups>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionUsageAlertGroups
 usn
```

***usn***

**Purpose:** Specifies the subscription associated with the usage alert groups.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getSubscriptionUsageAlertGroup()

```
SubscriptionUsageAlertGroup getSubscriptionUsageAlertGroup(string
 usn, string usageAlertGroupId)
```

This method returns information about the current state of a specified usage alert group. Use this method to retrieve and display information for a specific usage alert group for a subscription.

**Note:** You must enable pre-rating before you use this method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the subscription associated with the usage alert group.

**usageAlertGroupId**

**Format:** string

**Purpose:** Specifies the ID of the usage alert group for which you want to retrieve information.

#### Results

This method returns a SubscriptionUsageAlertGroup XML document that lists information about the current state of the usage alert group, for example, the usage alert group limit, amount consumed in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

#### Example: A returned SubscriptionUsageAlertGroup XML document

This example shows a SubscriptionUsageAlertGroup document for the Mobile Data Usage usage alert group. The subscription has exceeded the usage alert group limit for the current rating period.

```
<SubscriptionUsageAlertGroup xmlns="http://xml.inomial.com/
smile/2.xsd">
  <UsageAlertGroup>
    <usageAlertGroupID>3</usageAlertGroupID>
    <company>1</company>
    <name>Mobile Data Usage</name>
    <alertThresholds>
      <alertThreshold>50</alertThreshold>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </alertThresholds>
  </UsageAlertGroup>
  <periodEnd>2012-08-31+09:30</periodEnd>
  <limit>1000000</limit>
  <currentUse>211929000</currentUse>
  <currentThreshold>100</currentThreshold>
</SubscriptionUsageAlertGroup>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or usage alert group does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionUsageAlertGroup
 usn usageAlertGroupId
```

***usn***

**Purpose:** Specifies the subscription associated with the usage alert group.

***usageAlertGroupId***

**Purpose:** Specifies the ID of the usage alert group for which you want to retrieve information.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Retrieve spend limit states

A spend limit tracks the amount that a subscription spends on a particular type of usage in each rating period, and lets you take action when a customer exceeds spend limit alert thresholds. For example, a fixed-line subscription may have a spend limit of $150 on international calls per month. Smile can perform an action when a subscription exceeds the value of a spend limit, for example, disable international calls.

Smile resets the value of each spend limit to $0 at the start of each rating period and when a subscription changes plan.

Retrieving spend limit states lets you view information about the amount that a customer has spent on a particular type of usage in the current rating period.

The following methods retrieve spend limit states:

- `getSubscriptionSpendLimitStates()` returns information about all spend limits for a subscription.
- `getSubscriptionSpendLimitState()` returns information about a specified spend limit.

### `getSubscriptionSpendLimitStates()`

```
SubscriptionSpendLimitState getSubscriptionSpendLimitStates(string
 usn)
```

This method returns information about the current state of all spend limits associated with a subscription. Use this method to retrieve and display information about a subscription's spend limits.

**Note:** You must enable pre-rating before you use this method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the spend limits.

#### Results

This method returns a SubscriptionSpendLimitState XML document that lists information about the current state of each of the subscription's spend limits, for example, the spend limit value, amount spent in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

### Example: A returned SubscriptionSpendLimitState XML document

This example shows a SubscriptionSpendLimitState XML document for a subscription with two spend limits. The Data spend limit has a limit of $10 and the SMS spend limit has a limit of $50. The subscription has exceeded both spend limits.

```
<SubscriptionSpendLimitState xmlns="http://xml.inomial.com/
smile/2.xsd">
  <SubscriptionSpendLimit>
    <SpendLimit>
      <spendLimitId>1</spendLimitId>
      <sid>137</sid>
      <name>Data</name>
      <limit>10</limit>
      <alertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </alertThresholds>
    </SpendLimit>
    <periodEnd>2012-08-31+10:00</periodEnd>
    <effectiveLimit>10</effectiveLimit>
    <effectiveAlertThresholds>
      <alertThreshold>50</alertThreshold>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </effectiveAlertThresholds>
    <currentSpend>33.473000</currentSpend>
    <currentThreshold>100</currentThreshold>
    <previousThreshold>0</previousThreshold>
  </SubscriptionSpendLimit>
  <SubscriptionSpendLimit>
    <SpendLimit>
      <spendLimitId>7</spendLimitId>
      <sid>137</sid>
      <name>SMS</name>
      <limit>50</limit>
      <alertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </alertThresholds>
    </SpendLimit>
    <periodEnd>2012-08-31+10:00</periodEnd>
    <effectiveLimit>50</effectiveLimit>
    <effectiveAlertThresholds>
      <alertThreshold>50</alertThreshold>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </effectiveAlertThresholds>
    <currentSpend>56.178000</currentSpend>
    <currentThreshold>100</currentThreshold>
    <previousThreshold>50</previousThreshold>
  </SubscriptionSpendLimit>
</SubscriptionSpendLimitState>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionSpendLimitStates
 usn
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the spend limits.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSubscriptionSpendLimitState()`

```
SubscriptionSpendLimit getSubscriptionSpendLimitState(string usn,
 string spendLimitId)
```

This method returns information about the current state of a specified spend limit. Use this method to retrieve and display information about a specific spend limit for a subscription.

**Note:** You must enable pre-rating before you use this method.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the spend limit.

**spendLimitId**

**Format:** string

**Purpose:** Specifies the ID of the spend limit for which to retrieve information.

### Results

This method returns a SubscriptionSpendLimit XML document that lists information about the current state of the spend limit, for example, the spend limit value, amount spent in the period to date, alert thresholds and reset date.

**Note:** Smile updates the values in this XML document each time the pre-rating process runs.

There are no messages defined by this method.

### Example: A returned SubscriptionSpendLimit XML document

This example shows a SubscriptionSpendLimit XML document for the SMS spend limit. The subscription has exceeded the spend limit of $50.

```
<SubscriptionSpendLimit xmlns="http://xml.inomial.com/smile/2.xsd">
  <SpendLimit>
```

```
    <spendLimitId>7</spendLimitId>
    <sid>137</sid>
    <name>SMS</name>
    <limit>50</limit>
    <alertThresholds>
      <alertThreshold>50</alertThreshold>
      <alertThreshold>80</alertThreshold>
      <alertThreshold>100</alertThreshold>
    </alertThresholds>
  </SpendLimit>
  <periodEnd>2012-08-31+10:00</periodEnd>
  <effectiveLimit>50</effectiveLimit>
  <effectiveAlertThresholds>
    <alertThreshold>50</alertThreshold>
    <alertThreshold>80</alertThreshold>
    <alertThreshold>100</alertThreshold>
  </effectiveAlertThresholds>
  <currentSpend>56.178000</currentSpend>
  <currentThreshold>100</currentThreshold>
  <previousThreshold>50</previousThreshold>
</SubscriptionSpendLimit>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or spend limit does not exist.

`InvalidRequestException`

This fault is returned when the spend limit is not associated with the subscription.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionSpendLimitState usn
 spendLimitId
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the spend limit.

***spendLimitId***

**Purpose:** Specifies the ID of the spend limit for which to retrieve information.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Set spend limit overrides

Spend limit overrides let you change the default spend limit configuration for a subscription. You define the default spend limit configuration when you configure a service.

The following methods set spend limit overrides:

- `setSpendLimitOverride()` permanently changes the value of a spend limit.
- `setTemporarySpendLimitOverride()` changes the value of a spend limit for a specified rating period.
- `setSpendLimitAlertThresholdOverrides()` changes the alert thresholds for a spend limit.

### `setSpendLimitOverride()`

```
void setSpendLimitOverride(string usn,
 SubscriptionSpendLimitOverrideRequest
 subscriptionSpendLimitOverrideRequest)
```

This method permanently overrides the default value of a subscription spend limit. Use this method to permanently increase or decrease the value of a spend limit. If you have previously changed the value of a spend limit, you can also use this method to revert the spend limit to its default value.

Note that you cannot remove a spend limit from a subscription.

**Note:** You must enable pre-rating before you use this method.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the spend limit to change.

**subscriptionSpendLimitOverrideRequest**

**Format:** SubscriptionSpendLimitOverrideRequest XML

**Purpose:** Defines a SubscriptionSpendLimitOverrideRequest XML document that specifies the ID of the spend limit to change and the new value of the spend limit. If you do not specify an override value, Smile removes any existing overrides and reverts the spend limit value to the service default.

#### Example: Temporarily override a spend limit

This example changes the value of the spend limit with an ID of 7 to $10.

```
<SubscriptionSpendLimitOverrideRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <spendLimitId>7</spendLimitId>
  <limitOverride>10.00</limitOverride>
</SubscriptionSpendLimitOverrideRequest>
```

### Results

This method does not return any output, and changes the value of the specified spend limit.

There are no messages defined by this method. However, if the new spend limit value causes the subscription to cross an alert threshold, Smile emits the `SubscriptionSpendLimitStatusUpdated` message after the pre-rating process runs.

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

`InvalidRequestException`

This fault is returned when the spend limit does not exist or is not associated with the subscription, or an invalid value is provided for the new spend limit, for example, a negative number.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SetSpendLimitOverride usn
 spendLimitId [limitOverride]
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the spend limit to change.

***spendLimitId***

**Purpose:** Specifies the ID of the spend limit.

***limitOverride***

**Purpose:** (Optional) Specifies the new spend limit. If you do not specify this parameter, Smile removes any existing overrides and reverts the spend limit value to the service default.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `setTemporarySpendLimitOverride()`

```
void setTemporarySpendLimitOverride(string usn,
 TemporarySubscriptionSpendLimitOverrideRequest
 temporarySubscriptionSpendLimitOverrideRequest)
```

This method overrides the value of a subscription spend limit for a specified rating period. Use this method to temporarily increase or decrease the value of a spend limit. If you have previously changed the value of a spend limit, you can also use this method to temporarily revert the spend limit to its default value.

Note that you cannot remove a spend limit from a subscription.

**Note:** You must enable pre-rating before you use this method.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the spend limit to change.

**temporarySubscriptionSpendLimitOverrideRequest**

**Format:** TemporarySubscriptionSpendLimitOverrideRequest XML

**Purpose:** Defines a TemporarySubscriptionSpendLimitOverrideRequest XML document that specifies the spend limit ID, override value and rating period.

If you do not specify an override value and the spend limit has an existing permanent override, Smile temporarily removes the override and reverts the spend limit value to the service default.

If you do not specify an override value and the spend limit has an existing temporary override, Smile removes the temporary override and reverts the spend limit to the subscription-specific value. If no subscription-specific value is defined, Smile reverts the spend limit value to the service default.

### Example: Set a temporary spend limit override

This example changes the value of the spend limit with an ID of 7 to $10 in the current rating period only.

```
<TemporarySubscriptionSpendLimitOverrideRequest xmlns="http://
xml.inomial.com/smile/2.xsd">
  <spendLimitId>7</spendLimitId>
  <RatingPeriod>
    <currentPeriod/>
  </RatingPeriod>
  <limitOverride>10.00</limitOverride>
</TemporarySubscriptionSpendLimitOverrideRequest>
```

### Results

This method does not return any output, and changes the value of the specified spend limit.

There are no messages defined by this method. However, if the new spend limit value causes the subscription to cross an alert threshold, Smile emits the `SubscriptionSpendLimitStatusUpdated` message after the pre-rating process runs.

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

`InvalidRequestException`

This fault is returned when the spend limit does not exist, the spend limit or rating period is not associated with the subscription, or an invalid value is provided for the new spend limit, for example, a negative number.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SetTemporarySpendLimitOverride
 usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the spend limit to change.

***filename***

**Purpose:** Defines the filename of the TemporarySubscriptionSpendLimitOverrideRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `setSpendLimitAlertThresholdOverrides()`

```
void setSpendLimitAlertThresholdOverrides(string usn,
 SubscriptionSpendLimitAlertThresholdOverrideRequest
 subscriptionSpendLimitAlertThresholdOverrideRequest)
```

This method sets new alert thresholds for a subscription spend limit. If you have previously changed the spend limit alert thresholds for a subscription, you can also use this method to revert the alert thresholds to their default value.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription associated with the spend limit alert thresholds to change.

**subscriptionSpendLimitAlertThresholdOverrideRequest**

**Format:** SubscriptionSpendLimitAlertThresholdOverrideRequest XML

**Purpose:** Defines a SubscriptionSpendLimitAlertThresholdOverrideRequest XML document that specifies the ID of the spend limit and the new alert thresholds.

Smile removes any spend limit alert thresholds that are not specified in this document. If you do not specify any alert thresholds, Smile removes any existing overrides and reverts the alert thresholds to the service default.

### Example: Override a spend limit alert threshold

This example changes the alert thresholds for the spend limit with an ID of 7. The new alert thresholds are at 50%, 85% and 100%.

```
<SubscriptionSpendLimitAlertThresholdOverrideRequest xmlns="http://
xml.inomial.com/smile/2.xsd">
  <spendLimitId>7</spendLimitId>
  <AlertThresholdOverrides>
    <alertThreshold>50</alertThreshold>
    <alertThreshold>85</alertThreshold>
```

```
    <alertThreshold>100</alertThreshold>
  </AlertThresholdOverrides>
</SubscriptionSpendLimitAlertThresholdOverrideRequest>
```

### Results

This method does not return any output, and changes the value of the spend limit alert thresholds.

There are no messages defined by this method. However, if the new alert thresholds cause a subscription spend limit to cross an alert threshold, Smile emits the `SubscriptionSpendLimitStatusUpdated` message after the pre-rating process runs.

### Faults

`NoSuchItemException`

This fault is returned when the subscription does not exist.

`InvalidRequestException`

This fault is returned when the spend limit does not exist, the spend limit or rating period is not associated with the subscription, or an invalid value is provided for the new alert thresholds, for example, a negative number.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.SetSpendLimitAlertThresholdOverrides
 usn spendLimitId [threshold1 [threshold2...]]
```

***usn***

**Purpose:** Specifies the USN of the subscription associated with the spend limit alert thresholds to change.

***spendLimitId***

**Purpose:** Specifies the ID of the spend limit.

***threshold1 threshold 2...***

**Purpose:** (Optional) Specifies the space-separated list of new spend limit alert thresholds. Smile removes any alert thresholds that are not in this list. If you do not specify this parameter, Smile removes any existing overrides and reverts the alert thresholds to the service default.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Listen for spend limit changes

A spend limit restricts the amount that a subscription can spend on a particular usage type in each rating period. Each spend limit has one or more alert thresholds.

Smile emits a message on SmileSubscription when a subscription crosses an alert threshold. To help customers manage their monthly spend, you can write an application that listens for the messages and notifies a customer when they cross an alert threshold.

Smile emits the following spend limit messages when:

- `SubscriptionSpendLimitExceeded`—a subscription exceeds a spend limit.
- `SubscriptionSpendLimitReceded`—a subscription that had exceeded a spend limit is within the spend limit again, for example, the spend limit is reset at the start of the next rating period.
- `SubscriptionSpendLimitStatusUpdate`—a subscription crosses a spend limit alert threshold, or a spend limit is reset at the start of a rating period.

### Related reference

SmileSubscription on page 489

# Chapter 9

# Managing payments

## Payments API overview

Managing payments allows you to access payment information, calculate and process different types of payment and update and delete payment details. Payments can be processed once or payment details can be stored for future payments.

Methods that manage payments are located on the payment endpoint. The payment WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/payment?wsdl`

Related reference

Payment endpoint on page 443

## Retrieve payment information

Retrieving payment information lets you view the current state of a payment and the types of acceptable payments for a user.

The following methods retrieve payment information:

- `getPayment()` returns information about a payment for a specified payment number.
- `getPaymentTypes()` returns information about the payment methods for the current user.
- `getPendingPayments()` returns a list of all pending payments, including scheduled payments set up from payment plans, for a particular account.
- `getAutoPaymentOption()` returns the configured automatic payment option for an account.

### `getPayment()`

```
Payment getPayment(string paymentNumber)
```

This method returns information about a payment for a specified payment number. If the payment returned is in waiting status, its next transfer timestamp will be populated.

#### Parameters

**paymentNumber**

**Format:** string

**Purpose:** Specifies the payment number.

#### Results

This method returns information about a payment for a particular payment number.

There are no messages defined by this method.

#### Example: Payment XML document for a payment number that is a pending payment

This example shows a returned Payment XML document when payment number 21578596 is given as a parameter and it is a pending payment.

```
<Payment xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentNumber>21578596</PaymentNumber>
  <PaymentType>
    <Name>VISA</Name>
  </PaymentType>
  <Account>2142421094</Account>
  <Amount formatted="$1.00" currency="AUD">1.00</Amount>
  <Surcharge>
    <Amount formatted="$0.00" currency="AUD">0</Amount>
    <TaxAmount formatted="$0.00" currency="AUD">0</TaxAmount>
  </Surcharge>
  <Status>Waiting</Status>
  <NextTransfer>2014-08-06T10:28:46.703+10:00</NextTransfer>
```

```
</Payment>
```

### Example: Payment XML document for a payment number that is not a pending payment

This example shows a returned Payment XML document when payment number 21578612 is given as a parameter and it is not a pending payment.

```
<Payment xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentNumber>21578612</PaymentNumber>
  <PaymentType>
    <Name>Cash</Name>
  </PaymentType>
  <Account>2142421094</Account>
  <Amount formatted="$109.00" currency="AUD">109.00</Amount>
  <Surcharge>
    <Amount formatted="$0.00" currency="AUD">0</Amount>
    <TaxAmount formatted="$0.00" currency="AUD">0</TaxAmount>
  </Surcharge>
  <Status>Accepted</Status>
</Payment>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified payment number does not match a payment.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPayment paymentNumber
```

***paymentNumber***

**Purpose:** Specifies the payment number.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getPaymentTypes()

```
PaymentTypes getPaymentTypes(string currency)
```

This method returns information about the payment methods appropriate for the current user and the provided currency, and the surcharges for each payment method.

#### Parameters

**currency**

**Format:** string

**Purpose:** (Optional) Specifies the ISO 4217 currency code to filter payment type results. If not provided, payment types of all currencies will be returned.

#### Results

This method returns a PaymentTypes XML document that contains information about the payment methods a user may use. This document also lists any payment surcharges and the fields that must be completed to use a payment method.

There are no messages defined by this method.

#### Example: A returned PaymentTypes XML document

This example shows a returned PaymentTypes XML document. The payment type is American Express in Australian dollars. This payment method has a $0.50 fixed surcharge plus an additional 1% surcharge of the cost, with a minimum charge of $1.00.

```
<PaymentTypes xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <PaymentType>
    <Name>American Express</Name>
    <Currency>AUD</Currency>
    <Surcharge/>
    <FieldLabels>
      <Name>Card holder</Name>
      <Number>Card number</Number>
      <ExpiryDate>Expiry</ExpiryDate>
    </FieldLabels>
  </PaymentType>
  <PaymentTypeCode>payway</PaymentTypeCode>
    <Name>Westpac PayWay</Name>
    <Currency>AUD</Currency>
    <Surcharge>
      <Percentage>1</Percentage>
      <Fixed formatted="$0.50" currency="AUD">0.5</Fixed>
      <Minimum formatted="$1.00" currency="AUD">1</Minimum>
    </Surcharge>
    <FieldLabels>
      <Number>Number</Number>
      <CVV>CVV</CVV>
    </FieldLabels>
  </PaymentType>
```

```
</PaymentTypes>
```

### Faults

`NoSuchItemException`

This fault is returned when the currency code provided is not configured in Smile for the current company.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPaymentTypes [currency]
```

***currency***

**Purpose:** (Optional) Specifies the ISO 4217 currency code to filter payment type results. If not provided, payment types of all currencies will be returned.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getPendingPayments()`

```
PaymentListType getPendingPayments(string accountUsn)
```

This method returns a list of all pending payments, including scheduled payments set up from payment plans, for a particular account.

### Parameters

**accountUsn**

**Format:** string

**Purpose:** Specifies the account USN.

### Results

This method returns a list of all pending payments, including scheduled payments set up from payment plans, for a particular account USN.

There are no messages defined by this method.

### Example: A returned PaymentListType XML document

This example shows a returned PaymentListType XML document when account USN 2142425772 is given as a parameter.

```
<PaymentList xmlns:ns2="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <Payment>
    <PaymentNumber>21437017</PaymentNumber>
    <PaymentType>
      <Name>Bill Buddy</Name>
    </PaymentType>
    <Account>2142425772</Account>
    <Amount currency="AUD" formatted="$1.00">1.00</Amount>
    <Surcharge>
```

```
      <Amount currency="AUD" formatted="$0.00">0</Amount>
      <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
    </Surcharge>
    <Status>Waiting</Status>
    <NextTransfer>2016-02-17T13:14:31.482+11:00</NextTransfer>
    <PaymentDetails>
      <AccountName Disclosure="Full">test</AccountName>
      <AccountNumber Disclosure="Hint">1234 34</AccountNumber>
      <ExpiryDate Disclosure="Full">12/2017</ExpiryDate>
    </PaymentDetails>
    <ManuallyEntered>true</ManuallyEntered>
    <ManuallyDeclined>false</ManuallyDeclined>
    <AttemptCount>0</AttemptCount>
    <AttemptLogs/>
  </Payment>
  <Payment>
    <PaymentNumber>21437009</PaymentNumber>
    <PaymentType>
      <Name>test.onlineGateway</Name>
    </PaymentType>
    <Account>2142425772</Account>
    <Amount currency="AUD" formatted="$4.00">4.00</Amount>
    <Surcharge>
      <Amount currency="AUD" formatted="$0.00">0</Amount>
      <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
    </Surcharge>
    <Status>Waiting</Status>
    <NextTransfer>2016-02-19T00:00:00.000+11:00</NextTransfer>
    <PaymentDetails>
      <AccountName Disclosure="Full">Mr John Smith</AccountName>
      <ExpiryDate Disclosure="Full">2/2016</ExpiryDate>
    </PaymentDetails>
    <ManuallyEntered>false</ManuallyEntered>
    <ManuallyDeclined>false</ManuallyDeclined>
    <AttemptCount>0</AttemptCount>
    <AttemptLogs/>
  </Payment>
  <Payment>
    <PaymentNumber>21436985</PaymentNumber>
    <PaymentType>
      <Name>test.onlineGateway</Name>
    </PaymentType>
    <Account>2142425772</Account>
    <Amount currency="AUD" formatted="$2.00">2.00</Amount>
    <Surcharge>
      <Amount currency="AUD" formatted="$0.00">0</Amount>
      <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
    </Surcharge>
    <Status>Waiting</Status>
    <NextTransfer>2016-02-19T00:00:00.000+11:00</NextTransfer>
    <PaymentDetails>
      <AccountName Disclosure="Full">Mr John Smith</AccountName>
      <ExpiryDate Disclosure="Full">2/2016</ExpiryDate>
    </PaymentDetails>
    <ManuallyEntered>false</ManuallyEntered>
    <ManuallyDeclined>false</ManuallyDeclined>
    <AttemptCount>0</AttemptCount>
```

```
    <AttemptLogs/>
  </Payment>
</PaymentList>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not correspond to an existing account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPendingPayments accountUsn
```

***accountUsn***

**Purpose:** Specifies the account USN.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAutoPaymentOption()`

```
getAutoPaymentOptionResponse getAutoPaymentOption(string usn)
```

This method returns the configured automatic payment option for an account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to get the auto payment option from.

### Example: Request an accounts automatic payment option

This example requests the automatic payment option for account CN666.

```
<getAutoPaymentOption xmlns="http://xml.inomial.com/smile/2.xsd">
   <usn>CN666</usn>
</getAutoPaymentOption>
```

### Results

This method returns automatic payment option configured for the requested account

If the payment option returned is from Account Terms then `<fromAccountTerms>` will have value true, otherwise false.

There are no messages defined by this method.

### Example: A returned getAutoPaymentOptionResponse XML document

This example shows a returned getAutoPaymentOptionresponse XML document.

```
<getAutoPaymentOptionResponse xmlns:"http://xml.inomial.com/
smile/2.xsd">
   <AutoPaymentOption fromAccountTerms="false" key="2">Schedule
 Payment of Account Balance</AutoPaymentOption>
</getAutoPaymentOptionResponse>
```

### Faults

`InvalidRequestException`

This fault is returned when USN is not specified in the Request.

`NoSuchItemException`

This fault is returned when the specified USN does not correspond to an existing account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.getAutoPaymentOption usn
```

***usn***

**Purpose:** Specifies the USN of the account to get the auto payment option from.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage payments

Managing payments lets you calculate and process different types of payment and update the payment status.

Validating payments lets you authenticate credit card details and ensure the card is valid. Contact Inomial to help you configure this API functionality and set up the credit card validation module.

The following methods manage payments:

- `computeSurcharge()` calculates the surcharge payable on a payment of a specified amount.
- `processPayment()` processes a payment and queues the debits to be allocated to the payment.
- `validatePayment()` validates credit card details.
- `recordFailedPayment()` records failed payment to a Smile account.
- `reschedulePaymentWithNewDetails()` reschedules a payment to a specified time with specified new payment details.
- `cancelScheduledPayment()` cancels a payment.

### `computeSurcharge()`

```
SurchargeCalculation computeSurcharge(string usn, string
 paymentTypeCode, CurrencyValue amount, date timestamp)
```

This method calculates the surcharge payable on a payment of the given amount for a specified payment type at a specified time.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account against which the surcharge calculation should be made.

**paymentTypeCode**

**Format:** string

**Purpose:** Specifies the code of the intended payment type.

**amount**

**Format:** CurrencyValue

**Purpose:** Specifies the amount to be paid. If the currency attribute is specified it will be checked against the expected currency.

**timestamp**

**Format:** date

**Purpose:** (Optional) Specifies the time at which the payment is made. If not specified the current time is used.

### Results

This method returns a SurchargeCalculation XML document that list information about the surcharge and the itemised charges used to calculate the total amount charged.

There are no messages defined by this method.

### Example: A returned SurchargeCalculation XML document

This example shows a SurchargeCalculation XML document that calculates the surcharge on an amount of $0.90.

```
<SurchargeCalculation xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <Surcharge>
      <Percentage>1</Percentage>
      <Fixed formatted="$0.50" currency="AUD">0.5</Fixed>
      <Minimum formatted="$1.00" currency="AUD">1</Minimum>
  </Surcharge>
  <TotalAmount formatted="$2.00" currency="AUD">2.00</TotalAmount>
  <SurchargeTotal formatted="$1.10" currency="AUD">1.10</
SurchargeTotal>
  <SurchargeTax formatted="$0.10" currency="AUD">0.10</SurchargeTax>
  <SurchargeExTax formatted="$1.00" currency="AUD">1.00</
SurchargeExTax>
  <OriginalAmount formatted="$0.90" currency="AUD">0.90</
OriginalAmount>
</SurchargeCalculation>
```

### Faults

`NoSuchItemException`

This fault is returned when the objects represented by usn or paymentTypeCode cannot be found.

`InvalidRequestException`

This fault is returned when a required parameter is not provided, currency is provided and does not match the expected currency, or the amount is invalid, for example, zero or negative.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ComputeSurcharge usn
 paymentTypeCode amount [timestamp]
```

***usn***

**Purpose:** Specifies the USN of the account against which the surcharge calculation should be made.

***paymentTypeCode***

**Purpose:** Specifies the code of the intended payment type.

***amount***

**Purpose:** Specifies the amount to be paid. If the currency attribute is specified it will be checked against the expected currency.

***timestamp***

**Purpose:** (Optional) Specifies the time at which the payment is made. If not specified the current time is used.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `processPayment()`

```
PaymentResponse processPayment(string usn, PaymentRequest
 paymentRequest, AllocationRequest allocationRequest)
```

This method processes a payment, for example, contacting bank with details, and queues the debits to be allocated to the payment, pending the payment's success. No credit will be created if the payment is rejected.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account that the payment receipt is allocated to.

**paymentRequest**

**Format:** PaymentRequest XML

**Purpose:** Defines a PaymentRequest XML document that specifies the details of the payment.

**Note:** The payment amount is inclusive of surcharges. The API caller has to be aware of the surcharges, and provide the expected surcharge amount in `IncludesSurchargeAmount`. If surcharges are not present, or incorrect, an `InvalidRequestException` fault is returned.

**allocationRequest**

**Format:** AllocationRequest XML

**Purpose:** Defines an AllocationRequest XML document that specifies the details of where to allocate portions of the payment if it processes successfully.

Allocations may only be made to debits on the account provided, up to a total amount of payment, minus surcharge. May be null.

You can also use auto-allocation instead of providing transaction number(s). Auto-allocate functionality is only allowed for manual payments. `<AutoAllocate>` set to `true` automatically allocates a payment beginning with the oldest unpaid invoice on the account. `<AutoAllocate>` set to `false` will assign the payment amount to the account, but not allocate it to a specific invoice.

If `<AllocateTo>` elements are provided as well as `<AutoAllocate>` set to `true` then auto-allocation will occur and `<AllocateTo>` is ignored.

### Example: Request a payment

This example shows a PaymentRequest document containing details for a Payway payment.

**Note:** The usn, paymentRequest and allocationRequest parameters are all required to call `processPayment`.

```
<PaymentRequest>
  <PaymentDetails>
    <PaymentType>payway</PaymentType>
    <AccountNumber>1234567890</AccountNumber>
    <PaymentAmount>2.00</PaymentAmount>
    <IncludesSurchargeAmount>1.10</IncludesSurchargeAmount>
    <BranchNumber></BranchNumber>
    <ExpiryDate>2012-12</ExpiryDate>
    <CVV>123</CVV>
    <Properties>
      <Object></Object>
    </Properties>
  </PaymentDetails>
</PaymentRequest>
```

### Example: Request a payment for payment plan

This example shows a PaymentRequest document containing details for a Payway payment and specifying a payment plan number that a payment amount of $2 will be allocated to.

**Note:** The usn, paymentRequest and allocationRequest parameters are all required to call `processPayment`.

```
<PaymentRequest>
  <PaymentDetails>
    <PaymentType>payway</PaymentType>
    <AccountNumber>1234567890</AccountNumber>
    <PaymentAmount>2.00</PaymentAmount>
```

```
    <IncludesSurchargeAmount>1.10</IncludesSurchargeAmount>
    <BranchNumber></BranchNumber>
    <ExpiryDate>2012-12</ExpiryDate>
    <CVV>123</CVV>
    <Properties>
      <Object></Object>
    </Properties>
  </PaymentDetails>
  <PaymentPlan>1010</PaymentPlan>
</PaymentRequest>
```

### Example: Request a payment allocation

This example shows an AllocationRequest document to allocate an amount of $0.90 against transaction 21707609.

**Note:** The usn, paymentRequest and allocationRequest parameters are all required to call `processPayment`.

```
<AllocationRequest>
  <AllocateTo>
    <TransactionNumber>21707609</TransactionNumber>
    <AllocateAmount>0.90</AllocateAmount>
  </AllocateTo>
</AllocationRequest>
```

### Example: Request an auto-allocated payment

This example shows an AllocationRequest document with the `<AutoAllocate>` element set to `true`. Smile will allocate the payment to any debits on the account, starting with the oldest unpaid invoice, automatically.

**Note:** The usn, paymentRequest and allocationRequest parameters are all required to call `processPayment`.

```
<AllocationRequest>
  <AutoAllocate>true</AutoAllocate>
</AllocationRequest>
```

### Results

This method returns a PaymentResponse XML document that contains information about the status of the payment request.

Smile emits the `TransactionClosed` message when the payment is accepted. Smile emits the `TransactionAllocated` message when the payment is allocated.

### Example: A returned PaymentResponse XML document

This example shows a returned PaymentResponse XML document. The document contains information about the payment and payment allocation and timestamped confirmation of payment receipt and surcharge.

```
<PaymentResponse xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <Payment>
    <PaymentNumber>61383006</PaymentNumber>
    <PaymentType>
      <PaymentTypeId>16</PaymentTypeId>
      <Name>Westpac PayWay</Name>
      <PaymentTypeCode>payway</PaymentTypeCode>
    </PaymentType>
    <Account>1000000008</Account>
    <Amount formatted="$2.00" currency="AUD">2.00</Amount>
    <Surcharge>
      <Amount formatted="$1.10" currency="AUD">1.10</Amount>
      <TaxAmount formatted="$0.10" currency="AUD">0.10</TaxAmount>
    </Surcharge>
    <Status>Accepted</Status>
  </Payment>
  <Allocation>
    <AllocateTo>
      <TransactionNumber>61383105</TransactionNumber>
      <AllocateAmount formatted="$1.10" currency="AUD">1.10</
AllocateAmount>
    </AllocateTo>
    <AllocateTo>
      <TransactionNumber>21707609</TransactionNumber>
      <AllocateAmount formatted="$0.90" currency="AUD">0.90</
AllocateAmount>
    </AllocateTo>
  </Allocation>
  <Receipt>
    <version>2.0</version>
    <transactionType>Receipt</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>61383006</transactionNumber>
    <currency>AUD</currency>
    <amount>-2.00</amount>
    <accountType>1</accountType>
    <gstAmount>0.00</gstAmount>
    <entryTimestamp>2012-12-17T09:51:17.868+11:00</entryTimestamp>
    <formattedAmount currency="AUD" formatted="$(2.00)">-2.00</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
    <unallocatedAmount>0.00</unallocatedAmount>
    <enteredByUsn>2142423066</enteredByUsn>
    <logTimestamp>2012-12-17T09:51:17.870+11:00</logTimestamp>
    <glCode>083</glCode>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$0.00">0.00</formattedUnallocatedAmount>
```

```
  </Receipt>
  <Surcharge>
    <version>2.0</version>
    <transactionType>Surcharge</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>61383105</transactionNumber>
    <currency>AUD</currency>
    <amount>1.10</amount>
    <accountType>1</accountType>
    <gstAmount>0.10</gstAmount>
    <entryTimestamp>2012-12-17T09:51:17.868+11:00</entryTimestamp>
    <salesReportCategory>surchrg</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$1.10">1.10</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.10">0.10</
formattedGstAmount>
    <unallocatedAmount>0.00</unallocatedAmount>
    <enteredByUsn>2142423066</enteredByUsn>
    <logTimestamp>2012-12-17T09:51:18.367+11:00</logTimestamp>
    <glCode>200</glCode>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$0.00">0.00</formattedUnallocatedAmount>
  </Surcharge>
</PaymentResponse>
```

### Faults

`InvalidRequestException`

This fault is returned when PaymentRequest or AllocationRequest are invalid.

`NoSuchItemException`

This fault is returned when the USN, PaymentRequest or AllocationRequest refer to an object that cannot be found.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ProcessPayment usn
 paymentRequest allocationRequest
```

***usn***

**Purpose:** Specifies the USN of the account.

***paymentRequest***

**Purpose:** Defines the path to the XML document containing the payment request.

***allocationRequest***

**Purpose:** Defines the path to the XML document containing the allocation request.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### validatePayment()

```
PaymentDetailsValidationResponse validatePayment(
 PaymentDetailsValidationRequest paymentDetailsValidationRequest)
```

This method passes credit card details to an upstream payment gateway for validation. Before you use this method, contact Inomial and your upstream provider to help you configure your payment system.

#### Parameters

**paymentDetailsValidationRequest**

**Format:** PaymentDetailsValidationRequest XML

**Purpose:** Defines a PaymentDetailsValidationRequest XML document that specifies the payment details to validate. These details may vary between providers.

#### Example: Validate a payment

This example requests validation for a Mastercard. The fields in this document vary depending on your Smile configuration and upstream provider. The <Object> elements are configurable and let you provide any required data to your provider.

```
<PaymentDetailsValidationRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <PaymentDetails>
    <PaymentType>Mastercard</PaymentType>
    <AccountName>John Citizen</AccountName>
    <AccountNumber>4111 1111 1111 1111</AccountNumber>
    <ExpiryDate>2001-10</ExpiryDate>
    <CVV>813</CVV>
    <PaymentAmount>17.23</PaymentAmount>
    <Properties>
      <Object/>
    </Properties>
  </PaymentDetails>
  <Properties>
    <Object/>
  </Properties>
</PaymentDetailsValidationRequest>
```

#### Results

This method returns the verification result and may return a validation message with additional details. The method response varies between providers.

There are no messages defined by this method.

#### Example: A returned PaymentDetailsValidationResponse document

This example shows a failed payment validation. The validation message varies between providers; we recommend that your code parse only the validation result.

```
<PaymentDetailsValidationResponse xmlns="http://xml.inomial.com/
smile/2.xsd">
```

```
  <Valid>false</Valid>
  <ValidationMessage>The credit card has expired</ValidationMessage>
</PaymentDetailsValidationResponse>
```

### Faults

`PaymentDetailsValidationException`

This fault is returned when the payment details could not be validated.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ValidatePaymentDetails filename
```

***filename***

**Purpose:** Defines the filename of the PaymentDetailsValidationRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `recordFailedPayment()`

```
Payment recordFailedPayment(string usn, PaymentRequest
 paymentRequest, string message)
```

This method records failed payment to a Smile account.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to record the failed payment against.

**paymentRequest**

**Format:** PaymentRequest XML

**Purpose:** Defines a PaymentRequest XML document that specifies the bank details that the failed payment is associated with. StoredDetailsRequest is not supported, only PaymentDetails is permitted.

**message**

**Format:** string

**Purpose:** Records the reason for the failed payment.

### Example: A PaymentRequest XML document

This example shows a PaymentRequest XML document.

```
<PaymentRequest xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentDetails>
    <PaymentType>imported payment</PaymentType>
    <AccountName>Tester</AccountName>
```

```
    <AccountNumber>1000000008</AccountNumber>
    <PaymentAmount>4</PaymentAmount>
    <IncludesSurchargeAmount>0.0</IncludesSurchargeAmount>
    <BranchNumber></BranchNumber>
    <ExpiryDate>2013-12</ExpiryDate>
    <Properties>
      <Object></Object>
    </Properties>
  </PaymentDetails>
</PaymentRequest>
```

### Results

This method records a failed payment to Smile. A decline fee will occur if the payment type specified in the paymentRequest is configured to do so. The payment will appear as declined and be returned.

There are no messages defined by this method.

### Example: A returned Payment XML document

This example shows a returned Payment XML document with a declined payment status.

```
<Payment xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentNumber>21472493</PaymentNumber>
  <PaymentType>
      <Name>test.payment.recordFailedPayment</Name>
      <PaymentTypeCode>fetchtv</PaymentTypeCode>
  </PaymentType>
  <Account>2142420997</Account>
  <Amount formatted="$4.00" currency="AUD">4.00</Amount>
  <Surcharge>
      <Amount formatted="$0.00" currency="AUD">0</Amount>
      <TaxAmount formatted="$0.00" currency="AUD">0</TaxAmount>
  </Surcharge>
  <Status>Declined</Status>
</Payment>
```

### Faults

`InvalidRequestException`

This fault is returned when the paymentRequest contains invalid fields.

`NoSuchItemException`

This fault is returned when the account/subscription is not found.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RecordFailedPayment usn filename
 reason
```

***usn***

**Purpose:** Specifies the USN of the account to record the failed payment against.

***filename***

**Purpose:** Defines the filename of the paymentRequest XML document.

***reason***

**Purpose:** Specifies the reason for the failed payment.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### reschedulePaymentWithNewDetails()

```
Payment reschedulePaymentWithNewDetails(string paymentNumber,
 PaymentDetailsRequest request, date timestamp)
```

This method reschedules a payment to a specified time with specified new payment details, which can be either the latest added stored payment details of an account that this payment number belongs to or specific stored payment details.

**Note:** Payments can only be rescheduled if they are in Waiting, Declined or Attention status.

#### Parameters

**paymentNumber**

**Format:** string

**Purpose:** Specifies the payment number.

**request**

**Format:** PaymentDetailsRequest XML

**Purpose:** Defines a PaymentDetailsRequest XML document that specifies the payment details to reschedule the payment.

**timestamp**

**Format:** date

**Purpose:** (Optional) Specifies the time to which the payment will be rescheduled. If not specified, the next transfer date of the payment will be used.

#### Example: Request of latest stored payment details

This example shows a request of payment details, specified to use the latest stored payment details.

```
<PaymentDetailsRequest xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <DefaultStoredPaymentDetails/>
</PaymentDetailsRequest>
```

#### Example: Request of specific stored payment details

This example shows a request of payment details, specified to use particular stored payment details.

```
<PaymentDetailsRequest xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <StoredPaymentDetailsId>fcb75c9b-1a68-4383-a07c-a8e451c98882</
StoredPaymentDetailsId>
</PaymentDetailsRequest>
```

#### Results

This method returns information about a new payment that just has been rescheduled.

There are no messages defined by this method.

### Example: A returned Payment XML document

This example shows a returned Payment XML document for new payment number 21628284 that just has been rescheduled from the payment number given as a parameter.

```
<Payment xmlns:ns2="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <PaymentNumber>21628284</PaymentNumber>
  <PaymentType>
    <Name>VISA</Name>
  </PaymentType>
  <Account>2142500947</Account>
  <Amount currency="AUD" formatted="$1.00">1.00</Amount>
  <Surcharge>
    <Amount currency="AUD" formatted="$0.00">0</Amount>
    <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
  </Surcharge>
  <Status>Waiting</Status>
  <NextTransfer>2016-08-15T00:00:00.000+10:00</NextTransfer>
  <PaymentDetails>
    <AccountName Disclosure="Full">Test Visa</AccountName>
    <ExpiryDate Disclosure="Full">12/2017</ExpiryDate>
  </PaymentDetails>
  <ManuallyEntered>false</ManuallyEntered>
  <ManuallyDeclined>false</ManuallyDeclined>
  <AttemptCount>0</AttemptCount>
  <AttemptLogs/>
</Payment>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified payment number does not exist or the account has no stored payment details given as a parameter.

`InvalidRequestException`

This fault is returned when:

- the payment is not in the 'Waiting' or 'Attention' status
- there are no new payment details in paymentDetailsRequest
- more than one payment details are given

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ReschedulePaymentWithNewDetails
 paymentNumber filename [timestamp]
```

***paymentNumber***

**Purpose:** Specifies the payment number.

***filename***

**Purpose:** Defines the path to the PaymentDetailsRequest XML document.

***timestamp***

**Purpose:** (Optional) Specifies the time to which the payment will be rescheduled. If not specified, the next transfer date of the payment will be used.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `cancelScheduledPayment()`

```
Payment cancelScheduledPayment(string paymentNumber)
```

This method cancels a payment.

#### Parameters

**paymentNumber**

**Format:** string

**Purpose:** Specifies the payment number to cancel.

#### Results

This method returns information about the payment that has been cancelled.

There are no messages defined by this method.

#### Example: A returned Payment XML document

This example shows a returned Payment XML document for payment number 21627930 which was given as a parameter and has been cancelled.

```
<Payment xmlns:ns2="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <PaymentNumber>21627930</PaymentNumber>
  <PaymentType>
    <Name>VISA</Name>
  </PaymentType>
  <Account>2142500947</Account>
  <Amount currency="AUD" formatted="$1.00">1.00</Amount>
  <Surcharge>
    <Amount currency="AUD" formatted="$0.00">0</Amount>
    <TaxAmount currency="AUD" formatted="$0.00">0</TaxAmount>
  </Surcharge>
  <Status>Cancelled</Status>
  <PaymentDetails>
    <AccountName Disclosure="Full">Test Visa</AccountName>
    <ExpiryDate Disclosure="Full">12/2017</ExpiryDate>
  </PaymentDetails>
  <ManuallyEntered>false</ManuallyEntered>
  <ManuallyDeclined>true</ManuallyDeclined>
  <AttemptCount>0</AttemptCount>
  <AttemptLogs/>
```

```
</Payment>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified payment number does not exist.

`InvalidRequestException`

This fault is returned when the payment is not in the 'Waiting' or 'Attention' status.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CancelScheduledPayment
 paymentNumber
```

***paymentNumber***

**Purpose:** Specifies the payment number to cancel.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage stored payment details

Stored payment details records the information required to make a payment in Smile, so that future payments can be processed without re-entering payment details.

**Note:** If you wish to comply with PCI DSS requirements, Inomial recommends that you use a token-based payment gateway and only store tokens in Smile, so that access to Smile does not need to be controlled by stringent PCI DSS requirements.

Please note that although the API is built to support multiple sets of stored payment details per account, currently only 1 set of details per account is supported. This will change in the future.

Managing stored payments lets you access payment information and add, update and delete stored payment details.

The following methods manage stored payment details:

- `getStoredPaymentDetails()` returns the stored payment details for a specific ID on an account.
- `getStoredPaymentDetailsList()` returns all stored payment details for an account.
- `addStoredPaymentDetails()` adds stored payment details to an account.
- `updateStoredPaymentDetails()` updates the stored payment details on an account.
- `deleteStoredPaymentDetails()` removes stored payment details from an account.

### `getStoredPaymentDetails()`

```
StoredPaymentDetails getStoredPaymentDetails(string usn, string
 storedPaymentDetailsId)
```

This method returns the stored payment details for a given account and payment detail ID.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account of which to retrieve stored payment details.

**storedPaymentDetailsId**

**Format:** string

**Purpose:** Specifies the ID of the stored payment details to retrieve.

#### Results

This method returns a StoredPaymentDetails XML document containing the stored payment details. Certain fields may be concealed depending on Smile's configuration.

There are no messages defined by this method.

### Example: A returned StoredPaymentDetails XML document

This example shows a StoredPaymentDetails XML document with the stored payment details for a Visa card. Only a hint for the card number is displayed.

```
<StoredPaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <StoredPaymentDetailsId>4c47d515-8c02-4d1b-a416-1224b6979867</
StoredPaymentDetailsId>
  <AccountName Disclosure="Full">Test User</AccountName>
  <AccountNumber Disclosure="Hint">4111  44</AccountNumber>
  <ExpiryDate Disclosure="Full">12/2020</ExpiryDate>
  <PaymentType>
    <PaymentTypeCode>visa</PaymentTypeCode>
    <Name>Visa</Name>
    <Currency>AUD</Currency>
    <Surcharge/>
    <FieldLabels>
      <Name>Card holder</Name>
      <Number>Card number</Number>
      <ExpiryDate>Expiry</ExpiryDate>
    </FieldLabels>
  </PaymentType>
</StoredPaymentDetails>
```

### Faults

`NoSuchItemException`

This fault is returned when an account or stored payment details for the USN or stored payment details ID does not exist.

`InvalidRequestException`

This fault is returned when the USN and stored payment details ID do not correspond.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetStoredPaymentDetailsList usn
 storedPaymentDetailsId
```

***usn***

**Purpose:** Specifies the USN of the account of which to retrieve stored payment details.

***storedPaymentDetailsId***

**Purpose:** Specifies the ID of the stored payment details to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getStoredPaymentDetailsList()

```
StoredPaymentDetailsList getStoredPaymentDetailsList(string usn)
```

This method returns all stored payment details for a given account.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account of which to retrieve stored payment details.

#### Results

This method returns a StoredPaymentDetailsList XML document containing the stored payment details. Certain fields may be concealed depending on Smile's configuration.

There are no messages defined by this method.

#### Example: A returned StoredPaymentDetailsList XML document

This example shows a StoredPaymentDetailsList XML document with the stored payment details for a Visa card. Only a hint for the card number is displayed.

```
<StoredPaymentDetailsList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <StoredPaymentDetails>
    <StoredPaymentDetailsId>4c47d515-8c02-4d1b-a416-1224b6979867</
StoredPaymentDetailsId>
    <AccountName Disclosure="Full">Test User</AccountName>
    <AccountNumber Disclosure="Hint">4111  44</AccountNumber>
    <ExpiryDate Disclosure="Full">12/2020</ExpiryDate>
    <PaymentType>
      <PaymentTypeCode>visa</PaymentTypeCode>
      <Name>Visa</Name>
      <Currency>AUD</Currency>
      <Surcharge/>
      <FieldLabels>
        <Name>Card holder</Name>
        <Number>Card number</Number>
        <ExpiryDate>Expiry</ExpiryDate>
      </FieldLabels>
    </PaymentType>
  </StoredPaymentDetails>
</StoredPaymentDetailsList>
```

#### Faults

`NoSuchItemException`

This fault is returned when the account or stored payment details do not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
```

```
com.inomial.smile.client.v2.examples.GetStoredPaymentDetailsList usn
```

***usn***

**Purpose:** Specifies the USN of the account of which to retrieve stored payment details.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `addStoredPaymentDetails()`

```
StoredPaymentDetails addStoredPaymentDetails(string usn,
 PaymentDetails paymentDetails)
```

This method adds the stored payment details provided to the given account and returns the stored version, including stored payment ID.

Note that only one set of stored payment details per account is currently supported. If stored details already exist you must either delete them before calling this method or update them by calling `updateStoredPaymentDetails()`

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to add the payment details.

**paymentDetails**

**Format:** PaymentDetails XML

**Purpose:** Defines a PaymentDetails XML document that specifies the payment details to add.

#### Example: A PaymentDetails XML document

This example shows a PaymentDetails XML document to add a new Visa card to an account.

```
<PaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentType>visa</PaymentType>
  <AccountName>Test User</AccountName>
  <AccountNumber>4111111111111111</AccountNumber>
  <ExpiryDate>2020-12</ExpiryDate>
</PaymentDetails>
```

#### Example: A PaymentDetails XML document with a token

This example shows a PaymentDetails XML document to add a new Visa card to an account with a token.

```
<PaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd">
  <PaymentType>visa</PaymentType>
  <AccountName>John Citizen</AccountName>
  <Token hint="4222 ... 22">166570E5-D68F-4EEE-AF0F-4EEEED5FF493</
Token>
  <ExpiryDate>2020-12</ExpiryDate>
```

```
</PaymentDetails>
```

### Results

This method returns a StoredPaymentDetails XML document containing the stored payment details.

There are no messages defined by this method.

### Example: A returned StoredPaymentDetails XML document

This example shows a StoredPaymentDetails XML document with the stored payment details.

```
<StoredPaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <StoredPaymentDetailsId>d63d836e-55e2-4d0e-82b3-d148b368f712</
StoredPaymentDetailsId>
  <AccountName Disclosure="Full">Test User</AccountName>
  <AccountNumber Disclosure="Hint">4111  11</AccountNumber>
  <ExpiryDate Disclosure="Full">12/2020</ExpiryDate>
  <PaymentType>
    <PaymentTypeCode>visa</PaymentTypeCode>
    <Name>Visa</Name>
    <Currency>AUD</Currency>
    <Surcharge/>
    <FieldLabels>
      <Name>Card holder</Name>
      <Number>Card number</Number>
      <ExpiryDate>Expiry</ExpiryDate>
    </FieldLabels>
  </PaymentType>
</StoredPaymentDetails>
```

### Faults

`NoSuchItemException`

This fault is returned when the account does not exist.

`InvalidRequestException`

This fault is returned when payment details are invalid or missing.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddStoredPaymentDetailsList usn
 filename
```

***usn***

**Purpose:** Specifies the USN of the account to add the payment details.

***filename***

**Purpose:** Defines the filename of the PaymentDetails XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### updateStoredPaymentDetails()

```
StoredPaymentDetails updateStoredPaymentDetails(string usn, string
 storedPaymentDetailsId, PaymentDetails paymentDetails)
```

This method updates the stored payment details on a given account and returns the updated payment details.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to update.

**storedPaymentDetailsId**

**Format:** string

**Purpose:** Specifies the ID of stored payment details to update.

**paymentDetails**

**Format:** PaymentDetails XML

**Purpose:** Defines a PaymentDetails XML document that specifies the payment details to update.

#### Example: A PaymentDetails XML document

This example shows a PaymentDetails XML document to change the payment method to a Mastercard.

```
<PaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <PaymentType>mastercard</PaymentType>
  <AccountName>Test User</AccountName>
  <AccountNumber>5111222233334444</AccountNumber>
  <ExpiryDate>2022-12</ExpiryDate>
</PaymentDetails>
```

#### Results

This method returns a StoredPaymentDetails XML document containing the updated payment details.

There are no messages defined by this method.

#### Example: A returned StoredPaymentDetails XML document

This example shows a StoredPaymentDetails XML document with the updated payment details. The Mastercard number is fully concealed.

```
<StoredPaymentDetails xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <StoredPaymentDetailsId>d63d836e-55e2-4d0e-82b3-d148b368f712</
StoredPaymentDetailsId>
  <AccountName Disclosure="Full">Test User</AccountName>
  <ExpiryDate Disclosure="Full">12/2022</ExpiryDate>
```

```
  <PaymentType>
    <PaymentTypeCode>mastercard</PaymentTypeCode>
    <Name>Mastercard</Name>
    <Currency>AUD</Currency>
    <Surcharge/>
    <FieldLabels>
      <Name>Card holder</Name>
      <Number>Card number</Number>
      <ExpiryDate>Expiry</ExpiryDate>
    </FieldLabels>
  </PaymentType>
</StoredPaymentDetails>
```

### Faults

`NoSuchItemException`

This fault is returned when there is no account with the USN specified, or no payment details with the storedPaymentDetailsId exists.

`InvalidRequestException`

This fault is returned when payment details are invalid or missing, or the USN and stored payment details ID do not correspond.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateStoredPaymentDetailsList
 usn storedPaymentDetailsId filename
```

***usn***

**Purpose:** Specifies the USN of the account to update.

***storedPaymentDetailsId***

**Purpose:** Specifies the ID of the stored payment details to update.

***filename***

**Purpose:** Defines the filename of the PaymentDetails XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### deleteStoredPaymentDetails()

```
void deleteStoredPaymentDetails(string usn, string
 storedPaymentDetailsId)
```

This method deletes the stored payment details specified on a given account.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account to update.

**storedPaymentDetailsId**

**Format:** string

**Purpose:** Specifies the ID of stored payment details to delete.

#### Results

This method does not return any output for a successful deletion. An exception will be generated if the deletion attempt fails.

There are no messages defined by this method.

#### Faults

`NoSuchItemException`

This fault is returned when there is no account with the USN specified, or no payment details with the storedPaymentDetailsId exists.

`InvalidRequestException`

This fault is returned when the USN and stored payment details ID do not correspond.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.DeleteStoredPaymentDetailsList
 usn storedPaymentDetailsId
```

***usn***

**Purpose:** Specifies the USN of the account to update.

***storedPaymentDetailsId***

**Purpose:** Specifies the ID of the stored payment details to delete.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

# Chapter 10

# Managing prepaid

## Prepaid API overview

The prepaid API supports three types of prepaid offerings:

- **vouchers**—let customers validate and redeem prepaid vouchers using Smile's voucher management system
- **prepaid blocks**—let you add prepaid blocks to subscriptions without raising an invoice
- **self-serve prepaid**—let customers purchase prepaid on demand

Before you use the prepaid API, you must configure prepaid in Smile.

The prepaid WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/prepaid?wsdl`

### Related reference

Prepaid endpoint on page 450

# Use prepaid vouchers

A prepaid voucher lets customers buy and redeem prepaid at a time that suits them. For example, customers can use prepaid vouchers to access the internet at Smile wifi hotspots.

Vouchers are generated by Smile's voucher management system and contain a:

- **voucher number**—uniquely identifies the voucher across the company
- **secret code**—authenticates the voucher number
- **expiry date**—defines the length of time the voucher is valid

Before you use prepaid vouchers, you must configure the voucher management system and prepaid in Smile. You may need to complete the following tasks, among others:

- create and print vouchers
- create online sign-up forms
- configure prepaid in Smile
- configure connections between Smile, the network and walled gardens

A walled garden is an environment that limits the content and services that a prepaid user can access. For example, a walled garden may permit access to vendor-only content, such as a sign-up page.

The following methods use prepaid vouchers:

- `validateVoucher()` checks that the voucher number and secret code are valid, and that the voucher has not expired.
- `redeemVoucher()` consumes the voucher and adds prepaid to a subscription.

## Voucher use cases

Vouchers let customers buy and redeem prepaid at a time that suits them. The following examples demonstrate how the Smile API lets you write applications that use prepaid vouchers.

### Example: Add prepaid to an existing user

The following workflow lets an existing customer redeem a prepaid voucher:

1. `authenticate()` checks the customer's credentials and returns the subscription USN.

   **Note:** This method is located on the authentication endpoint.
2. `redeemVoucher()` consumes the voucher and adds prepaid to the specified USN.

### Example: Add prepaid to a new user

The following workflow lets a new customer redeem a prepaid voucher:

1. `validateVoucher()` checks that the voucher is valid.

   This step ensures that Smile does not create an account for an invalid voucher.
2. `orderAccount()` creates a new account and subscription.

3. `authenticate()` checks the new account's credentials and returns the subscription USN.
4. `redeemVoucher()` consumes the voucher and adds prepaid to the specified USN.

### `validateVoucher()`

```
VoucherValidationResponse validateVoucher(int company, string
 voucherNumber, string voucherCode)
```

This method verifies that a voucher is valid, but does not consume the voucher.

To be valid, a voucher must have a valid voucher number and secret code, and not have expired.

#### Parameters

**company**

**Format:** integer

**Purpose:** Specifies the identification number for the company that issued the voucher.

**Default:** 1

**voucherNumber**

**Format:** string

**Purpose:** Specifies the voucher number.

**voucherCode**

**Format:** string

**Purpose:** Specifies the voucher's secret code.

#### Results

This method validates the voucher and returns a VoucherValidationResponse XML document with the validation response.

There are no messages defined by this method.

#### Example: A returned VoucherValidationResponse XML document

This example shows a returned VoucherValidationResponse XML document. The voucher is not valid because the expiry date has passed.

```
<VoucherValidationResponse xmlns="http://xml.inomial.com/
smile/2.xsd">
  <valid>false</valid>
  <reason>The voucher has expired</reason>
</VoucherValidationResponse>
```

#### Faults

`PrepaidException`

This fault is returned when an error occurs or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ValidateVoucher company voucherNumber
 voucherCode
```

***company***

**Purpose:** Specifies the identification number for the company that issued the voucher.

***voucherNumber***

**Purpose:** Specifies the voucher number.

***voucherCode***

**Purpose:** Specifies the voucher's secret code.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `redeemVoucher()`

```
void redeemVoucher(string usn, string voucherNumber, string
 voucherCode)
```

This method validates a voucher and, if the voucher is valid, consumes the voucher and adds the prepaid value of the voucher to a subscription.

Before you call this method, you may need to call `authenticate()` to check the customer's credentials and obtain the subscription's USN.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to add the prepaid to.

**voucherNumber**

**Format:** string

**Purpose:** Specifies the voucher number.

**voucherCode**

**Format:** string

**Purpose:** Specifies the voucher's secret code.

### Results

This method adds the value of the voucher to the subscription. Depending on your Smile configuration and the value of the prepaid, the subscription's features may be updated after the prepaid is added, for example, the subscription may be unshaped or unwall-gardened. Smile may also reload the subscription's billing configuration and communicate with other network applications.

This method does not return any output. Smile does not emit any messages after it adds prepaid to the subscription, but if the prepaid causes the subscription's features to change, Smile emits the `SubscriptionFeaturesChanged` message when the feature change is complete.

### Faults

`PrepaidException`

This fault is returned when the voucher is invalid, the subscription cannot accept the voucher or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RedeemVoucher usn voucherNumber
 voucherCode
```

***usn***

**Purpose:** Specifies the USN of the subscription to add the prepaid to.

***voucherNumber***

**Purpose:** Specifies the voucher number.

***voucherCode***

**Purpose:** Specifies the voucher's secret code.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Use prepaid blocks

Prepaid blocks add prepaid to a subscription without raising an invoice or charging the customer.

Before you use prepaid blocks, you must configure prepaid in Smile.

The following methods use prepaid blocks:

- `getPrepaidState()` searches by USN for all purchased prepaid blocks and returns information about their current state.
- `addPrepaid()` adds prepaid to a subscription.
- `updatePrepaid()` updates a purchased prepaid block on a subscription.

### `getPrepaidState()`

```
SubscriptionPrepaidState getPrepaidState(string usn)
```

This method returns a list all prepaid blocks ever purchased for a USN. This may include prepaid blocks which have expired or have been completely consumed.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to query.

#### Results

This method returns a SubscriptionPrepaidState XML document contains a list of the prepaid blocks that have been added to the subscription.

There are no messages defined by this method.

**CAUTION:** Check the start and end date of the prepaid blocks to ensure that it is valid when displaying to users.

#### Example: A returned SubscriptionPrepaidState XML document

This example shows a SubscriptionPrepaidState XML document for subscription “2142429550” with three prepaid blocks in the history, one of which is completely consumed.

```
<SubscriptionPrepaidState xmlns:="http://xml.inomial.com/smile/2.xsd"
 xmlns:="http://ws.inomial.com/smile.2">
  <USN>2142429550</USN>
  <PrepaidStateElements>
    <PrepaidStateElement>
      <StartDate>2013-10-24T14:06:40.281+11:00</StartDate>
      <EndDate>2014-10-23T23:59:59.999+11:00</EndDate>
      <PrepaidDescription key="test1">Test Prepaid 1</
PrepaidDescription>
      <PurchasedQuantity formatted="0:10:00">600</PurchasedQuantity>
      <RemainingQuantity formatted="0:10:00">600</RemainingQuantity>
      <ChargeType key="0">Online time</ChargeType>
```

```
    </PrepaidStateElement>
    <PrepaidStateElement>
      <StartDate>2013-10-24T14:10:04.099+11:00</StartDate>
      <EndDate>2014-10-23T23:59:59.999+11:00</EndDate>
      <PrepaidDescription key="jfy">Prepaid Description</
PrepaidDescription>
      <PrepaidType key="295">new type</PrepaidType>
      <PurchasedQuantity formatted="1">1</PurchasedQuantity>
      <RemainingQuantity formatted="1">1</RemainingQuantity>
      <VoucherNumber>2188</VoucherNumber>
      <ChargeType key="0">Online time</ChargeType>
    </PrepaidStateElement>
    <PrepaidStateElement>
      <StartDate>2013-08-23T00:00:00.000+10:00</StartDate>
      <EndDate>2013-11-22T23:59:59.999+11:00</EndDate>
      <PrepaidDescription key="pp529">Block A</PrepaidDescription>
      <PurchasedQuantity formatted="0:04:00">240</PurchasedQuantity>
      <RemainingQuantity formatted="0:00:00">0</RemainingQuantity>
      <ChargeType key="887">test.prepaid.getPrepaidState Timed</
ChargeType>
    </PrepaidStateElement>
  </PrepaidStateElements>
</SubscriptionPrepaidState>
```

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPrepaidState usn
```

***usn***

**Purpose:** Specifies the USN of the subscription to query.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `addPrepaid()`

```
void addPrepaid(string usn, string prepaidCode,
 AddPrepaidRequestOverride addPrepaidRequestOverride)
```

This method adds a prepaid block to a subscription, but does not raise an invoice or charge for the prepaid block. This functionality lets you add prepaid to a subscription without using vouchers. Before you use this method, you must define prepaid blocks in Smile.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to which the prepaid block is added.

**prepaidCode**

**Format:** string

**Purpose:** Specifies the code of the prepaid block to add. You can view the prepaid code when you view the prepaid block configuration in Smile.

**addPrepaidRequestOverride**

**Format:** AddPrepaidRequestOverride XML

**Purpose:** (Optional) Defines an AddPrepaidRequestOverride XML document that specifies the fields to override.

### Example: AddPrepaidRequestOverride XML document

This example shows an AddPrepaidRequestOverride XML document for the prepaid block being purchased. The usage of the block has been set to be unlimited and it is expired at midnight of 25th Oct 2013.

```
<AddPrepaidRequestOverride xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Quantity unlimited="true">0</Quantity>
  <ExpiryDate>2013-10-25T00:00:00+11:00</ExpiryDate>
</AddPrepaidRequestOverride>
```

### Results

This method adds the value of the block to the subscription. Depending on your Smile configuration and the value of the prepaid, the subscription's features may be updated after the prepaid is added, for example, the subscription may be unshaped or unwall-gardened. Smile may also reload the subscription's billing configuration and communicate with other network applications.

If AddPrepaidRequestOverride XML is provided, the value of the block will be overridden accordingly, for example, expiry date and purchased quantity. For quantity, if the 'unlimited' attribute is set to true, the quantity value will be ignored. In terms of expiry, either expiry date time or expiry duration can be specified.

This method does not return any output.

Smile emits the `PrepaidAdded` message after it adds prepaid to the subscription. If the prepaid causes the subscription's features to change, Smile emits the `SubscriptionFeaturesChanged` message when the feature change is complete.

### Faults

`PrepaidException`

This fault is returned when the prepaid code is invalid, the subscription does not support the prepaid block or the caller does not have permission to perform the action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddPrepaid usn prepaidCode
 [filename]
```

***usn***

**Purpose:** Specifies the USN of the subscription to which the prepaid block is added.

***prepaidCode***

**Purpose:** Specifies the code of the prepaid block to add.

***fileaname***

**Purpose:** (Optional) Defines the filename of the AddPrepaidRequestOverride XML document that contains the fields to override.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `updatePrepaid()`

```
void updatePrepaid(string usn, PrepaidUpdate prepaidUpdate)
```

This method updates a purchased prepaid block from a subscription, but does not raise an invoice or charge. Before you use this method, there must be a block associated with the subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription whose prepaid block to update.

**prepaidUpdate**

**Format:** PrepaidUpdate XML

**Purpose:** Defines a PrepaidUpdate XML document that specifies the fields to update.

#### Example: A PrepaidUpdate XML document

This example shows a PrepaidUpdate XML document.

If no values are specified for the optional elements in PrepaidUpdate no changes will be applied to these fields. The original values will not be updated. A PrepaidUpdate XML document contains the following elements:

| | |
|---|---|
| **`<PrepaidId>`** | The ID of the purchased prepaid to update. |
| **`<StartDate>`** | (Optional) The start timestamp of the prepaid block. |
| **`<EndDate>`** | (Optional) The end timestamp of the prepaid block. |
| **`<PurchasedQuantity>`** | (Optional) The purchased prepaid quantity.<br>An empty value indicates that the prepaid block is unlimited. Therefore, an end date must be present. |
| **`<RemainingQuantity>`** | (Optional) The remaining prepaid quantity. |

```
<PrepaidUpdate xmlns="http://xml.inomial.com/smile/2.xsd">
  <PrepaidId>474</PrepaidId>
  <StartDate>2015-02-05T00:00:00+11:00</StartDate>
  <EndDate>2015-02-25T00:00:00+11:00</EndDate>
  <PurchasedQuantity>500000000</PurchasedQuantity>
  <RemainingQuantity>400000000</RemainingQuantity>
```

```
</PrepaidUpdate>
```

### Results

This method does not return any output.

Smile emits the `PrepaidUpdated` message when a prepaid block is updated.

### Faults

`PrepaidException`

This fault is returned when:

- the USN provided does not exist
- no prepaid ID given in the PrepaidUpdate XML document
- the prepaid block to update does not belong to the subscription provided
- the end date is before the start date due to the update
- the end date is not provided when updating the purchased quantity with an empty value
- a remaining quantity is not provided when updating remaining quantity
- the remaining quantity is larger than the purchased quantity
- the purchased quantity is smaller than total used quantity

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdatePrepaid usn filename
```

***usn***

**Purpose:** Specifies the USN of the subscription whose prepaid block to update.

***filename***

**Purpose:** Defines the filename of the PrepaidUpdate XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Use self-serve prepaid

Self-serve prepaid lets customers purchase prepaid on demand.

Before you use self-serve prepaid, you must complete the following tasks in Smile, among others:

- configure prepaid
- make prepaid blocks available for self-service
- set up payment types

The following self-serve prepaid workflow lets you dynamically discover available prepaid blocks and payment types, so you can change your prepaid offerings in Smile without changing your code:

1. `authenticate()` checks the customer's credentials and returns the subscription USN.

   **Note:** This method is located on the authentication endpoint.
2. `getSelfServePrepaidList()` returns a list of the prepaid blocks a customer can purchase.
3. `getPrepaidPaymentTypes()` returns a list of payment methods available to the customer.
4. `getQuote()` returns a quote to purchase a prepaid block with a specific payment method.
5. `addSelfServePrepaid()` adds a prepaid block to a subscription and charges for the prepaid.

### `getSelfServePrepaidList()`

```
SelfServePrepaidList getSelfServePrepaidList(string usn)
```

This method returns a list of the prepaid blocks a subscription is eligible to purchase.

For example, the prepaid blocks available may vary by duration or download limit, and the prepaid blocks a subscription is eligible to buy may vary by business unit, account or location.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to return the available prepaid blocks for.

#### Results

This method returns a SelfServePrepaidList XML document with information about the prepaid a subscription can purchase, such as the rate, currency and code of each block.

There are no messages defined by this method.

#### Example: A returned SelfServePrepaidList XML document

This example shows a returned SelfServePrepaidList XML document with one available prepaid block.

```
<SelfServePrepaidList xmlns="http://xml.inomial.com/smile/2.xsd">
  <PrepaidDescription>
    <PrepaidCode>1 hr</PrepaidCode>
    <Description>1 hour block</Description>
```

```
    <DisplayAmount>$2.23</DisplayAmount>
    <Rate>2.2314</Rate>
    <Currency>AUD</Currency>
  </PrepaidDescription>
</SelfServePrepaidList>
```

### Faults

`PrepaidException`

This fault is returned when the subscription does not support self-serve prepaid, or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSelfServePrepaidList usn
```

***usn***

**Purpose:** Specifies the USN of the subscription to return the available prepaid blocks for.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getPrepaidPaymentTypes()`

```
PaymentTypes getPrepaidPaymentTypes(string usn, string prepaidCode)
```

This method returns information about the payment methods a customer can use to purchase a specific prepaid block, and the surcharges for each payment method.

Payment methods may vary because a customer may only be able to buy a particular block with local currency, or may not be able to use a credit card with a high surcharge to buy a discounted block, among other examples.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to return the available payment methods .

**prepaidCode**

**Format:** string

**Purpose:** Specifies the code of the prepaid block.

### Results

This method returns a PaymentTypes XML document that contains information about the payment methods a customer can use to purchase a prepaid block. The document also lists payment surcharges and the fields that must be completed to use a payment method.

There are no messages defined by this method.

### Example: A returned PaymentTypes XML document

This example shows a returned PaymentTypes XML document with a single payment type. This payment method has a $5.00 fixed surcharge plus an additional 5% surcharge of the prepaid cost. For example, if this payment method is used to buy a $10 prepaid block, the surcharge is $5.50. Note that the fields in this document vary by payment type.

```
<PaymentTypes xmlns="http://xml.inomial.com/smile/2.xsd">
  <PaymentType>
    <PaymentTypeCode>visa</PaymentTypeCode>
    <Name>Visa</Name>
    <Currency>AUD</Currency>
    <Surcharge>
      <Percentage>5.00</Percentage>
      <Fixed>5.00</Fixed>
      <Minimum>0.00</Minimum>
    </Surcharge>
    <FieldLabels>
      <Name>Name on card</Name>
      <Number>Card number</Number>
      <CVV>CVV</CVV>
      <ExpiryDate>Expiry</ExpiryDate>
    </FieldLabels>
  </PaymentType>
</PaymentTypes>
```

### Faults

`PrepaidException`

This fault is returned when the subscription does not support the prepaid block or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPrepaidPaymentTypes usn
 prepaidCode
```

***usn***

**Purpose:** Specifies the USN of the subscription to return the available payment methods.

***prepaidCode***

**Purpose:** Specifies the code of the prepaid block.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getQuote()`

```
ItemisedTransactionDetail getQuote(string usn, string prepaidCode,
 string paymentTypeCode)
```

This method returns a quote for the cost of a prepaid block purchased with a specific payment method. The quote may differ from the value returned by the `getPrepaidPaymentTypes()` method, for example, if the subscription is entitled to a discount. The method also returns the information the customer must provide to purchase the prepaid.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to return the quote for.

**prepaidCode**

**Format:** string

**Purpose:** Specifies the code of the prepaid block.

**paymentTypeCode**

**Format:** string

**Purpose:** Specifies the code of the payment type.

#### Results

This method returns an ItemisedTransactionDetail XML document that lists the cost to purchase the prepaid block and the information the customer must provide to buy the block.

There are no messages defined by this method.

#### Example: A returned ItemisedTransactionDetail XML document

This example shows an ItemisedTransactionDetail XML document that lists the cost to buy a single prepaid block. The block costs $15 and there is an additional GST cost of $1.50, for a total cost of $16.50.

```
<ItemisedTransactionDetail xmlns="http://xml.inomial.com/
smile/2.xsd">
  <version>2.0</version>
  <transactionType>Quote</transactionType>
  <company>1</company>
  <usn>2142420823</usn>
  <transactionNumber>21433586</transactionNumber>
  <currency>AUD</currency>
  <amount>16.50</amount>
  <accountType>1</accountType>
  <gstAmount>1.50</gstAmount>
  <salesReportCategory>invoice</salesReportCategory>
  <formattedAmount currency="AUD" formatted="$16.50">16.50</
formattedAmount>
  <formattedGstAmount currency="AUD" formatted="$1.50">1.50</
formattedGstAmount>
  <unallocatedAmount>16.50</unallocatedAmount>
```

```
  <formattedUnallocatedAmount currency="AUD"
 formatted="$16.50">16.50</formattedUnallocatedAmount>
  <transactionDate>2012-06-28+10:00</transactionDate>
  <openDate>2012-06-28+10:00</openDate>
  <quoteNumber>21433586</quoteNumber>
  <createdByUsn>2142423595</createdByUsn>
  <transactionItem>
    <lineNumber>0</lineNumber>
    <itemCode>006100</itemCode>
    <description>Prepaid time purchased</description>
    <quantity>1</quantity>
    <amount>15.00</amount>
    <isGst>false</isGst>
    <chargeGst>false</chargeGst>
    <count>0</count>
    <gstAmount>1.50</gstAmount>
  </transactionItem>
  <transactionItem>
    <lineNumber>1</lineNumber>
    <itemCode>gst</itemCode>
    <description>Includes 10% GST</description>
    <quantity>1</quantity>
    <amount>1.50</amount>
    <isGst>false</isGst>
    <chargeGst>false</chargeGst>
    <count>0</count>
    <gstAmount>0.00</gstAmount>
  </transactionItem>
</ItemisedTransactionDetail>
```

### Faults

`PrepaidException`

This fault is returned when the subscription does not support the prepaid block, the payment type is not available for the prepaid block or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetQuote usn prepaidCode paymentTypeCode
```

***usn***

**Purpose:** Specifies the USN of the subscription to return the quote for.

***prepaidCode***

**Purpose:** Specifies the code of the prepaid block.

***paymentTypeCode***

**Purpose:** Specifies the code of the payment type.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### addSelfServePrepaid()

```
void addSelfServePrepaid(string usn, string prepaidCode,
 PaymentRequest paymentRequest)
```

This method adds a prepaid block to a subscription and charges for the prepaid.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to add the prepaid block to.

**prepaidCode**

**Format:** string

**Purpose:** Specifies the identifier of the prepaid block to add to the subscription.

**paymentRequest**

**Format:** PaymentRequest XML

**Purpose:** Defines a PaymentRequest XML document that specifies payment information to purchase the prepaid.

#### Example: Purchase a prepaid block

This example uses a Mastercard to buy a prepaid block worth $16.50.

```
<PaymentRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <PaymentDetails>
    <PaymentType>mastercard</PaymentType>
    <AccountName>Joe Smith</AccountName>
    <AccountNumber>411111111111</AccountNumber>
    <ExpiryDate>2012-12</ExpiryDate>
    <CVV>123</CVV>
    <PaymentAmount>16.50</PaymentAmount>
  </PaymentDetails>
  <Object />
</PaymentRequest>
```

#### Results

This method adds the prepaid value of the block to the subscription. Depending on your Smile configuration and the value of the prepaid, the subscription's features may be updated after the prepaid is added, for example, the subscription may be unshaped or unwall-gardened. Smile may also reload the subscription's billing configuration and communicate with other network applications.

This method does not return any output. Smile does not emit any messages after it adds prepaid to the subscription, but if the prepaid causes the subscription's features to change, Smile emits the `SubscriptionFeaturesChanged` message when the feature change is complete.

#### Faults

`PrepaidException`

This fault is returned when the subscription does not support the prepaid block, the PaymentRequest XML document does not match the prepaid code (for example, an incorrect amount is specified) or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddSelfServePrepaid usn
 prepaidCode filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to add the prepaid block to.

***prepaidCode***

**Purpose:** Specifies the identifier of the prepaid block to add to the subscription.

***filename***

**Purpose:** Defines the filename of the PaymentRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

# Chapter 11

# Managing contracts

## Contract API overview

A contract specifies the length of time that a customer must remain on a plan, for example, 12 or 24 months. Each contract belongs to a specific subscription.

A contract may include setup charges, termination charges and payment method restrictions. Unless otherwise defined, the total contract value is the sum of all the recurring charges for the duration of the contract, plus any setup charges. It does not include usage charges.

The contract WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/contract?wsdl`

### Related reference

Contract endpoint on page 428

## Retrieve contract information

Retrieving contract information lets you view the availability of contracts for services, subscriptions and plans and the contracts currently in use.

The following methods retrieve contract information:

- `getContractsForService()` returns a list of contracts for a service.
- `getActiveSubscriptionContract()` returns information about the contract a subscription is bound to at a given time.
- `getAvailableContractsForPlan()` returns a list of contracts available to a subscription on a particular plan at a given time.
- `getCurrentSubscriptionContract()` returns the current contract for a subscription.
- `getSubscriptionContractById()` returns a subscription contract.
- `getSubscriptionContracts()` returns a list of all contracts for a subscription.

### `getContractsForService()`

```
ContractDescriptionList getContractsForService(int sid)
```

This method returns a list of contracts defined against a particular service. This can be useful for presenting a set of available contracts to an end user. Note that there are other variables which may constrain the contract available to a subscription such as the plan.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the ID of the service whose contracts to retrieve.

#### Results

This method returns a ContractDescriptionList XML document containing information about available contracts for the given service.

There are no messages defined by this method.

#### Example: A returned ContractDescriptionList XML document

This example shows a ContractDescriptionList XML document containing the available contracts for service 414.

```
<ContractDescriptionList xmlns="http:// xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/ smile.2">
  <ContractDescription>
    <ContractId>465</ContractId>
    <ServiceId key="414">Spend Wise Voice Service</ServiceId>
    <Name>24 Months</Name>
    <BaseValue currency="AUD" formatted="$500.00">500</BaseValue>
    <PlanRecurringChargePercentage>0</PlanRecurringChargePercentage>
    <Duration unit="CalendarMonths">24</Duration>
```

```
    <GraceDays>2</GraceDays>
    <ValidFrom>2014-08-21+10:00</ValidFrom>
    <ValidTo>2014-08-21+10:00</ValidTo>
    <SetupFees/>
    <EarlyTerminationFee>
      <ItemCode>002595</ItemCode>
      <Name>Early termination charge</Name>
      <UnitSize>1</UnitSize>
      <ChargeGst>true</ChargeGst>
      <Rate currency="AUD" formatted="$50.00">50</Rate>
    </EarlyTerminationFee>
    <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
    <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
    <AllowedManagedPlanChanges>All</AllowedManagedPlanChanges>
  </ContractDescription>
  <ContractDescription>
    <ContractId>464</ContractId>
    <ServiceId key="414">Spend Wise Voice Service</ServiceId>
    <Name>12 Months</Name>
    <BaseValue currency="AUD" formatted="$100.00">100</BaseValue>
    <PlanRecurringChargePercentage>0</PlanRecurringChargePercentage>
    <Duration unit="CalendarMonths">12</Duration>
    <GraceDays>2</GraceDays>
    <ValidFrom>2014-08-21+10:00</ValidFrom>
    <SetupFees/>
    <EarlyTerminationFee>
      <ItemCode>002594</ItemCode>
      <Name>Early termination charge</Name>
      <UnitSize>1</UnitSize>
      <ChargeGst>true</ChargeGst>
      <Rate currency="AUD" formatted="$12.00">12</Rate>
    </EarlyTerminationFee>
    <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
    <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
    <Comments>Asd</Comments>
    <AllowedManagedPlanChanges>All</AllowedManagedPlanChanges>
  </ContractDescription>
</ContractDescriptionList>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided SID does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetContractsForService sid
```

***sid***

**Purpose:** Specifies the ID of the service whose contracts to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### getActiveSubscriptionContract()

```
SubscriptionContract getActiveSubscriptionContract(string usn,
 dateTime when)
```

This method returns details about the contract a given subscription is bound to (if any) at a given time.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to query.

**when**

**Format:** dateTime

**Purpose:** Specifies the time stamp of when to query the active contract.

#### Results

This method returns a SubscriptionContract XML document that contains information about the contract a given subscription is bound to.

There are no messages defined by this method.

#### Example: A returned SubscriptionContract XML document

This example shows a SubscriptionContract XML document containing information about contract ID 476 that subscription 2142422589 is bound to.

```
<SubscriptionContract xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContractId>476</SubscriptionContractId>
  <Contract key="399">12 Month</Contract>
  <USN>2142422589</USN>
  <EarlyTerminationCharge currency="AUD" formatted="$0.00">0</
EarlyTerminationCharge>
  <StartDate>2014-08-28+10:00</StartDate>
  <EndDate>2015-08-27+10:00</EndDate>
  <ContractValue currency="AUD" formatted="$360.00">360.000000</
ContractValue>
  <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
  <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
</SubscriptionContract>
```

#### Faults

`InvalidRequestException`

This fault is returned when the USN or when parameter is not provided.

`NoSuchItemException`

This fault is returned when the subscription USN does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetActiveSubscriptionContract
 usn when
```

***usn***

**Purpose:** Specifies the USN of the subscription to query.

***when***

**Purpose:** Specifies the time stamp of when to query the active contract.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAvailableContractsForPlan()`

```
PlanContractDescriptionList getAvailableContractsForPlan(int company,
 int plan, dateTime when)
```

This method returns contracts available to subscriptions on a particular plan at a particular time. This can be useful for presenting a set of contracts to an end user which may be available for use for an existing subscription already subscribed to a particular plan.

If no contracts are available an empty list is returned.

### Parameters

**company**

**Format:** integer

**Purpose:** Specifies the ID of the company to query.

**plan**

**Format:** integer

**Purpose:** Specifies the code of the plan to query.

**when**

**Format:** dateTime

**Purpose:** Specifies the time stamp at which the contracts must be valid for use.

### Results

This method returns a PlanContractDescriptionList XML document that contains information about the available contracts for a given company and plan.

If no contracts are available an empty list is returned.

There are no messages defined by this method.

### Example: A returned PlanContractDescriptionList XML document

This example shows a PlanContractDescriptionList XML document containing information about available contracts.

```
<PlanContractDescriptionList xmlns="http://xml.inomial.com/
smile/2.xsd" xmlns="http://ws.inomial.com/smile.2">
  <PlanContractDescription>
    <PlanContractId>558</PlanContractId>
    <SelfService>false</SelfService>
    <Name>Example contract</Name>
    <Plan key="standard">Silver</Plan>
    <ContractDescription>
      <ContractId>464</ContractId>
      <ServiceId key="414">Spend Wise Voice Service</ServiceId>
      <Name>12 Months</Name>
      <BaseValue currency="AUD" formatted="$100.00">100</BaseValue>
      <PlanRecurringChargePercentage>0</
PlanRecurringChargePercentage>
      <Duration unit="CalendarMonths">12</Duration>
      <GraceDays>2</GraceDays>
      <ValidFrom>2014-08-21+10:00</ValidFrom>
      <SetupFees/>
      <EarlyTerminationFee>
        <ItemCode>002594</ItemCode>
        <Name>Early termination charge</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$12.00">12</Rate>
      </EarlyTerminationFee>
      <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
      <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
      <Comments>Asd</Comments>
      <AllowedManagedPlanChanges>All</AllowedManagedPlanChanges>
      </Plans>
    </ContractDescription>
  </PlanContractDescription>
</PlanContractDescriptionList>
```

### Faults

`NoSuchItemException`

This fault is returned when the given plan does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAvailableContractsForPlan
 company plan when
```

***company***

**Purpose:** Specifies the ID of the company to query.

***plan***

**Purpose:** Specifies the code of the plan to query.

***when***

**Purpose:** Specifies the time stamp at which the contracts must be valid for use.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getCurrentSubscriptionContract()`

```
SubscriptionContract getCurrentSubscriptionContract(string usn)
```

This method returns the current contract for a subscription.

The method is equivalent to calling `getActiveSubscriptionContract()` providing the current time stamp to the when parameter.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to query.

#### Results

This method returns a SubscriptionContract XML document containing information about the current contract for a given subscription.

There are no messages defined by this method.

#### Example: A returned SubscriptionContract XML document

This example shows a SubscriptionContract XML document containing information about the current contract for subscription 2142422589.

```
<SubscriptionContract xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContractId>476</SubscriptionContractId>
  <Contract key="399">12 Month</Contract>
  <USN>2142422589</USN>
  <EarlyTerminationCharge currency="AUD" formatted="$0.00">0</
EarlyTerminationCharge>
  <StartDate>2014-08-28+10:00</StartDate>
  <EndDate>2015-08-27+10:00</EndDate>
  <ContractValue currency="AUD" formatted="$360.00">360.000000</
ContractValue>
  <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
  <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
</SubscriptionContract>
```

### Faults

`NoSuchItemException`

This fault is returned when the provided USN does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCurrentSubscriptionContract usn
```

***usn***

**Purpose:** Specifies the USN of the subscription to query.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSubscriptionContractById()`

```
SubscriptionContract getSubscriptionContractById(string
 subscriptionContractId)
```

This method returns a subscription contract by its ID.

### Parameters

**subscriptionContractId**

**Format:** string

**Purpose:** Specifies the ID of the subscription contract to retrieve.

### Results

This method returns a SubscriptionContract XML document containing information about the subscription contract.

There are no messages defined by this method.

### Example: A returned SubscriptionContract XML document

This example shows a SubscriptionContract XML document for subscription contract 476.

```
<SubscriptionContract xmlns="http:// xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContractId>476</SubscriptionContractId>
  <Contract key="399">12 Month</Contract>
  <USN>2142422589</USN>
  <EarlyTerminationCharge currency="AUD" formatted="$0.00">0</
EarlyTerminationCharge>
  <StartDate>2014-08-28+10:00</StartDate>
  <EndDate>2015-08-27+10:00</EndDate>
  <ContractValue currency="AUD" formatted="$360.00">360.000000</
ContractValue>
  <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
  <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
```

```
</SubscriptionContract>
```

### Faults

`ItemNotFoundException`

This fault is returned when the given subscription contract ID does not exist.

`InvalidRequestException`

This fault is returned when the request is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionContractById
 subscriptionContractID
```

***subscriptionContractID***

**Purpose:** Specifies the ID of the subscription contract to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSubscriptionContracts()`

```
SubscriptionContractList getSubscriptionContracts(string usn)
```

This method returns a lists of all contracts for a given subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription whose contracts to return.

### Results

This method returns a SubscriptionContractList XML document containing information about all contracts associated with a subscription.

There are no messages defined by this method.

### Example: A returned SubscriptionContractList XML document

This example shows a SubscriptionContractList XML document containing information about contracts associated with subscription 2142422431.

```
<SubscriptionContractList xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContract>
    <SubscriptionContractId>474</SubscriptionContractId>
    <Contract key="465">24 Months</Contract>
    <USN>2142422431</USN>
    <EarlyTerminationCharge currency="AUD" formatted="$50.00">50</
EarlyTerminationCharge>
```

```
    <StartDate>2014-08-21+10:00</StartDate>
    <EndDate>2016-08-20+10:00</EndDate>
    <ContractValue currency="AUD" formatted="$500.00">500</
ContractValue>
    <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
    <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
    <Terminated>2014-08-22+10:00</Terminated>
    <ContractedPlanScheduleItem>1201</ContractedPlanScheduleItem>
  </SubscriptionContract>
  <SubscriptionContract>
    <SubscriptionContractId>475</SubscriptionContractId>
    <Contract key="465">24 Months</Contract>
    <USN>2142422431</USN>
    <EarlyTerminationCharge currency="AUD" formatted="$50.00">50</
EarlyTerminationCharge>
    <StartDate>2014-08-23+10:00</StartDate>
    <EndDate>2015-08-22+10:00</EndDate>
    <ContractValue currency="AUD" formatted="$500.00">500</
ContractValue>
    <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
    <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
    <Terminated>2014-09-30+10:00</Terminated>
    <ContractedPlanScheduleItem>1201</ContractedPlanScheduleItem>
  </SubscriptionContract>
  <SubscriptionContract>
    <SubscriptionContractId>473</SubscriptionContractId>
    <Contract key="464">12 Months</Contract>
    <USN>2142422431</USN>
    <EarlyTerminationCharge currency="AUD"
 formatted="$120,000.00">120000</EarlyTerminationCharge>
    <StartDate>2020-08-21+10:00</StartDate>
    <EndDate>2030-08-20+10:00</EndDate>
    <ContractValue currency="AUD" formatted="$100.00">100</
ContractValue>
    <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
    <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
    <ContractedPlanScheduleItem>1201</ContractedPlanScheduleItem>
  </SubscriptionContract>
</SubscriptionContractList>
```


### Faults

`NoSuchItemException`

This fault is returned when the USN provided does not exist.

`InvalidRequestException`

This fault is returned when the request is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSubscriptionContracts usn
```

***usn***

**Purpose:** Specifies the USN of the subscription whose contracts to return.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage contracts

Managing contracts let you add, update and remove contracts.

The following methods manage contracts:

- `addSubscriptionContract()` adds a contract to subscription.
- `updateSubscriptionContract()` updates the values of a contract on a subscription.
- `proposeContractTermination()` prepares a quote of charges for a contract termination.
- `commitContractTermination()` performs a contract termination.

### `addSubscriptionContract()`

```
SubscriptionContract addSubscriptionContract(string usn, int
 contractId, dateTime startDate, dateTime endDate)
```

This method adds a contract to a subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription.

**contractId**

**Format:** integer

**Purpose:** Specifies the ID of the contract to add.

**startDate**

**Format:** dateTime

**Purpose:** Specifies the start date and time of the contract.

**endDate**

**Format:** dateTime

**Purpose:** (Optional) Specifies the end date, and therefore duration, of the contract. The contract will end at the end of the specified day (not the start).

#### Results

This method returns a SubscriptionContract XML document that contains information about the contract.

There are no messages defined by this method.

#### Example: A returned SubscriptionContract XML document

This example shows a SubscriptionContract XML document. Contract ID 557 has been added to subscription 2142422431.

```
<SubscriptionContract xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
```

```
  <SubscriptionContractId>557</SubscriptionContractId>
  <Contract key="465">24 Months</Contract>
  <USN>2142422431</USN>
  <EarlyTerminationCharge currency="AUD" formatted="$50.00">50</
EarlyTerminationCharge>
  <StartDate>2032-08-20+10:00</StartDate>
  <EndDate>2034-08-19+10:00</EndDate>
  <ContractValue currency="AUD" formatted="$500.00">500</
ContractValue>
  <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
  <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
  <ContractedPlanScheduleItem>1201</ContractedPlanScheduleItem>
</SubscriptionContract>
```

### Faults

`NoSuchItemException`

This fault is returned when the USN or contract ID do not exist.

`InvalidContractException`

This fault is returned when the subscription's plan does not support the contract or if the contract start date overlaps another contract.

`InvalidRequestException`

This fault is returned when the request is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddSubscriptionContract usn
 contractId startDate endDate
```

***usn***

**Purpose:** Specifies the USN of the subscription.

***contractId***

**Purpose:** Specifies the ID of the contract to add.

***startDate***

**Purpose:** Specifies the start date and time of the contract.

***endDate***

**Purpose:** (Optional) Specifies the end date of the contract.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### updateSubscriptionContract()

```
void updateSubscriptionContract(string usn, SubscriptionContract
 subscriptionContract)
```

This method updates the given subscription contract with new values.

Note that many values of contracts are immutable. You can only update the following parameters:

- comments
- contract value
- early termination charge
- start date
- end date
- maximum termination charge
- minimum termination charge

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription to update.

**subscriptionContract**

**Format:** SubscriptionContract XML

**Purpose:** Defines a SubscriptionContract XML document that specifies the contract values to update.

#### Example: Update a subscription contract

This example shows a request to update subscription contract 476 for subscription 2142422589.

```
<SubscriptionContract xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContractId>476</SubscriptionContractId>
  <Contract key="399">12 Month</Contract>
  <USN>2142422589</USN>
  <EarlyTerminationCharge currency="AUD" formatted="$0.00">0</
EarlyTerminationCharge>
  <StartDate>2014-08-28+10:00</StartDate>
  <EndDate>2015-08-27+10:00</EndDate>
  <ContractValue currency="AUD" formatted="$360.00">360.000000</
ContractValue>
  <MinimumTerminationCharge currency="AUD" formatted="$0.00">0</
MinimumTerminationCharge>
  <MaximumTerminationCharge currency="AUD" formatted="$0.00">0</
MaximumTerminationCharge>
</SubscriptionContract>
```

#### Results

This method does not return any output.

Smile emits the `RatingConfigurationUpdated` message when the subscription contract is updated.

### Faults

`ContractOverlapException`

This fault is returned when changing the start or end date causes the contract to overlap with another contract.

`NoSuchItemException`

This fault is returned when the contract or subscription does not exist.

`InvalidContractException`

This fault is returned when an attempt is made to modify an immutable property of a contract, or if the contract ID or USN elements is missing.

`InvalidRequestException`

This fault is returned when the request is invalid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateSubscriptionContract usn
 filename
```

***usn***

**Purpose:** Specifies the USN of the subscription to update.

***filename***

**Purpose:** Defines the filename of the SubscriptionContract XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### proposeContractTermination()

```
SubscriptionContractTerminationOffer
 proposeContractTermination(string usn, int subscriptionContractId,
 dateTime terminationDate)
```

This method is used to propose a contract termination to Smile. In response, Smile will return an offer detailing the charges which would be raised due to the contract termination without actually terminating the contract. The proposal may then be relayed to `commitContractTermination()` , with possible modifications to the resultant invoice, which will carry out the contract termination.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to propose the contract termination.

**subscriptionContractId**

**Format:** integer

**Purpose:** Specifies the ID of the contract to propose termination of.

**terminationDate**

**Format:** dateTime

**Purpose:** Specifies the date and time at which the proposed termination would take effect.

#### Results

This method returns a SubscriptionContractTerminationOffer XML document that contains information about any charges that would be raised in the event of the contract termination.

There are no messages defined by this method.

#### Example: A retured SubscriptionContractTerminationOffer XML document

This example shows a SubscriptionContractTerminationOffer XML document detailing an early termination charge that would be raised if contract 557 was terminated at the given date.

```
<SubscriptionContractTerminationOffer xmlns="http://xml.inomial.com/
smile/ 2.xsd" xmlns="http://ws.inomial.com/smile.2">
  <OfferId>46488a01-e97c-3a6d-a354-115ae954fa1d</OfferId>
  <SubscriptionContractId>557</SubscriptionContractId>
  <USN>2142422431</USN>
  <TerminationDate>2032-12-28+11:00</TerminationDate>
  <NewInvoiceRequest>
    <effectiveDate>2015-02-09+11:00</effectiveDate>
    <dueDate>2015-02-21+11:00</dueDate>
    <ChargeRequest>
      <USN>2142422431</USN>
      <itemCode>002595</itemCode>
      <description>Early termination charge</description>
      <count>1</count>
      <startDate>2032-12-28+11:00</startDate>
      <endDate>2032-12-29+11:00</endDate>
```

```
      <amount>460.30</amount>
      <chargeGst>true</chargeGst>
      <chargeId>46488a01-e97c-3a6d-a354-115ae954fa1d</chargeId>
    </ChargeRequest>
  </NewInvoiceRequest>
</SubscriptionContractTerminationOffer>
```

### Faults

`NoSuchItemException`

This fault is returned when the contract or subscription is not found.

`InvalidRequestException`

This fault is returned when:

- the given subscription contract ID is not associated with the given USN
- the termination date is beyond the end date of the contract
- the contract has already terminated
- the termination date is null

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ProposeContractTermination usn
 subscriptionContractdId terminationDate
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to propose the contract termination.

***subscriptionContractId***

**Purpose:** Specifies the ID of the contract to propose termination of.

***terminationDate***

**Purpose:** Specifies the date and time at which the proposed termination would take effect.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `commitContractTermination()`

```
SubscriptionContractTerminationResponse
 commitContractTermination(string usn,
 SubscriptionContractTerminationOffer
 subscriptionContractTerminationOffer)
```

This method is used to terminate a contract. This method accepts an offer previously provided by Smile to the client. The offer includes details about when the contract is to be terminated as well as the charges to be raised. If the charges are not present then no termination charges are raised. If the offer ID is present in the offer document, Smile checks that the offer is still valid. If the offer is no longer valid an `OfferExpiredException` is thrown.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to terminate the contract.

**subscriptionContractTerminationOffer**

**Format:** SubscriptionContractTerminationOffer XML

**Purpose:** Defines a SubscriptionContractTerminationOffer XML document that specifies the details of the contract termination.

#### Example: Request a contract termination

This example submits contract termination offer 46488a01-e97c-3a6d-a354-115ae954fa1d for processing.

```
<SubscriptionContractTerminationOffer>
  <OfferId>46488a01-e97c-3a6d-a354-115ae954fa1d</OfferId>
  <SubscriptionContractId>557</SubscriptionContractId>
  <USN>2142422431</USN>
  <TerminationDate>2032-12-28+11:00</TerminationDate>
  <NewInvoiceRequest>
    <effectiveDate>2015-02-09+11:00</effectiveDate>
    <dueDate>2015-02-21+11:00</dueDate>
    <ChargeRequest>
      <USN>2142422431</USN>
      <itemCode>002595</itemCode>
      <description>Early termination charge</description>
      <count>1</count>
      <startDate>2032-12-28+11:00</startDate>
      <endDate>2032-12-29+11:00</endDate>
      <amount>460.30</amount>
      <chargeGst>true</chargeGst>
      <chargeId>46488a01-e97c-3a6d-a354-115ae954fa1d</chargeId>
    </ChargeRequest>
  </NewInvoiceRequest>
</SubscriptionContractTerminationOffer>
```

### Results

This method returns a SubscriptionContractTerminationResponse XML document that contains information about the contract termination.

Smile emits the `ItemisedTransactionClosed` message when the termination has been processed.

### Example: A returned SubscriptionContractTerminationResponse XML document

This example shows a SubscriptionContractTerminationResponse XML document.


```
<SubscriptionContractTerminationResponse xmlns="http://
xml.inomial.com/smile/2.xsd" xmlns="http://ws.inomial.com/smile.2">
  <SubscriptionContractId>557</SubscriptionContractId>
  <USN>2142422431</USN>
  <TerminationDate>2032-12-28+11:00</TerminationDate>
  <Invoice>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>2142422423</usn>
    <transactionNumber>45040516</transactionNumber>
    <currency>AUD</currency>
    <amount>506.33</amount>
    <accountType>1</accountType>
    <gstAmount>46.03</gstAmount>
    <entryTimestamp>2015-02-09T14:47:03.771+11:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$506.33">506.33</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$46.03">46.03</
formattedGstAmount>
    <unallocatedAmount>506.33</unallocatedAmount>
    <enteredByUsn>2142422415</enteredByUsn>
    <logTimestamp>2015-02-09T14:47:03.794+11:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$506.33">506.33</formattedUnallocatedAmount>
    <transactionDate>2015-02-09+11:00</transactionDate>
    <dueDate>2015-02-21+11:00</dueDate>
    <openDate>2015-02-09+11:00</openDate>
    <closeDate>2015-02-09+11:00</closeDate>
    <quoteNumber>45040516</quoteNumber>
    <createdByUsn>2142422415</createdByUsn>
    <discount>0.00</discount>
    <transactionSource>Manual</transactionSource>
    <transactionItem>
      <usn>2142422431</usn>
      <lineNumber>0</lineNumber>
      <itemCode>002595</itemCode>
      <chargeFrom>2032-12-28+11:00</chargeFrom>
      <chargeTo>2032-12-29+11:00</chargeTo>
      <description>Early termination charge</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>460.30</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
```

```
      <discount>0.00</discount>
      <count>1</count>
      <gstAmount>46.030000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142422431</usn>
      <lineNumber>1</lineNumber>
      <itemCode>gst</itemCode>
      <chargeFrom>2015-02-09+11:00</chargeFrom>
      <chargeTo>2015-02-09+11:00</chargeTo>
      <description>Includes 10% GST</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>46.03</amount>
      <isGst>true</isGst>
      <chargeGst>false</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>gst</salesReportCategory>
    </transactionItem>
  </Invoice>
</SubscriptionContractTerminationResponse>
```

### Faults

`NoSuchItemException`

This fault is returned when the subscription or contract is not found.

`InvalidRequestException`

This fault is returned when the request is invalid.

`OfferExpiredException`

This fault is returned when the offer ID was provided and a change made to the subscription, or its contract since the offer was generated has invalidated the offer, or the contract is already terminated.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CommitContractTermination usn
 filename
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to terminate the contract.

***filename***

**Purpose:** Defines the filename of the SubscriptionContractTerminationOffer XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Chapter

# 12

# Managing custom fields

## Custom fields API overview

The custom fields endpoint if the SOAP API lets you retrieve details of and manipulate entries of custom collections in Smile.

**Note:** All custom field methods assume that the custom collection already exists. You cannot define a new custom collection or delete a custom collection though SOAP. A custom collection must be created through the Smile UI.

Smile emits SmileCustomCollection messages after it completes an action.

The custom fields WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/config/customfields?wsdl`

### Related reference

Custom fields endpoint on page 434

## Retrieve custom field information

Retrieving custom field entries gets you information about the entries in a custom collection.

The following methods retrieve custom collection entries:

- `getCollectionEntry()` retrieves a custom collection entry directly by key.
- `getCollectionEntryByEntryCode()` retrieves details of a custom collection entry by entry code.
- `findCollectionEntries()` searches a custom collection custom field entries that match a given value.

### `getCollectionEntry()`

```
CustomCollectionEntry getCollectionEntry(int key)
```

This method retrieves a custom collection entry directly by key.

#### Parameters

**key**

**Format:** integer

**Purpose:** Specifies the primary identifier of the custom collection entry to be returned.

#### Results

This method returns a CustomCollectionEntry XML document describing the requested custom collection entry.

There are no messages defined by this method.

#### Example: A returned CustomCollectionEntry XML document

This example shows CustomCollectionEntry XML document.

```
<CustomCollectionEntry xmlns="http://xml.inomial.com/smile/2.xsd"
 key="6550">
  <CollectionCode>AirFreightDestinations</CollectionCode>
  <EntryCode>AU_Sydney</EntryCode>
  <Object objectId="3455" formId="125" displayName="SYD - Kingsford
 Smith International">
     <String name="airportName">Kingsford Smith International</String>
     <String name="airportIcaoCode">YSSY</String>
     <String name="airportIataCode">SYD</String>
  </Object>
</CustomCollectionEntry>
```

#### Faults

`NoSuchItemException`

This fault is returned when a custom collection entry does not exist with the given key.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCollectionEntry key
```

***key***

**Purpose:** Specifies the primary identifier of the custom collection entry to be returned.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getCollectionEntryByEntryCode()`

```
CustomCollectionEntryType getCollectionEntryByEntryCode(string
 collectionCode, string entryCode)
```

This method retrieves details of a custom collection entry by entry code.

### Parameters

**collectionCode**

**Format:** string

**Purpose:** Specifies the custom collection that the entry to be retrieved belongs to.

**entryCode**

**Format:** string

**Purpose:** Specifies the custom collection entry to be retrieved.

### Results

This method returns a CustomCollectionEntry XML document that describes the custom collection entry.

There are no messages defined by this method.

### Example: A returned CustomCollectionEntry XML document

This example shows a CustomCollectionEntry XML document.

```
<CustomCollectionEntry xmlns="http://xml.inomial.com/smile/2.xsd"
 key="6547">
  <CollectionCode>AirFreightDestinations</CollectionCode>
  <EntryCode>UK_London</EntryCode>
  <Object objectId="1266" formId="125" displayName="LHR - Heathrow">
    <String name="airportName">Heathrow</String>
    <String name="airportIcaoCode">EGLL</String>
    <String name="airportIataCode">LHR</String>
  </Object>
</CustomCollectionEntry>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified collection code does not exist, or if the collection lacks an entry with the given entry code.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetCollectionByEntryCode
 collectionCode entryCode
```

***collectionCode***

**Purpose:** Specifies the custom collection that the entry to be retrieved belongs to.

***entryCode***

**Purpose:** Specifies the custom collection entry to be retrieved.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `findCollectionEntries()`

```
CustomCollectionEntrySearchResults findCollectionEntries(string
 collectionCode, SearchCondition criteria, int limit, int offset)
```

This method searches a custom collection custom fields entries that match a given value. Alternatively this method can be used to retrieve a list of all custom collection entries by specifying an empty SearchCondition.

### Parameters

**collectionCode**

**Format:** string

**Purpose:** Specifies the custom collection to be searched.

**criteria**

**Format:** SearchCondition XML

**Purpose:** Defines a SearchCondition XML document that specifies the criterion for the return set. A special case of providing an empty `<SearchConditionType>` element will result in all entries of the given collection returned.

**limit**

**Format:** integer

**Purpose:** Specifies the maximum number of results to return. May not exceed 200.

**offset**

**Format:** integer

**Purpose:** Specifies the offset into the result set of where to begin returning results.

Note that the limit and offset parameters work in a similar way to SQL's LIMIT and OFFSET clauses. The result set order is guaranteed to be stable and non-repeating so that multiple consecutive calls to `findCollectionEntries()` can be made with offset=offset+limit to scroll through the result set.

### Example: A request SearchCondition XML document

This example shows a SearchCondition XML document.

```
<SearchCondition xmlns="http://ws.inomial.com/smile.2">
  <Equals>
    <Attribute>colour</Attribute>
    <Value>Purple</Value>
  <Equals>
</SearchCondition>
```

### Results

This method returns a CustomCollectionEntrySearchResults XML document containing a list of entries in the given collection that match the criteria. An empty `<CustomCollectionEntryList>` element inside the `<CustomCollectionEntrySearchResults>` element will be returned if there are zero matches.

If there are more results in the result set than the limit argument, then the moreResults attribute will be set to `true`. If you are at the end of the results set, then moreResults will be set to `false`.

There are no messages defined by this method.

### Example: A returned CustomCollectionEntrySearchResults XML document

This example shows a CustomCollectionEntrySearchResults XML document.

```
<CustomCollectionEntrySearchResults xmlns="http://ws.inomial.com/
smile.2" limit="200" offset="0" moreResults="false">
  <CustomCollectionEntryList>
    <CustomCollectionEntry key="6326">
      <CollectionCode>productList</CollectionCode>
      <EntryCode>modem</EntryCode>
      <Object objectId="6326" formId="870" displayName="adslsmodem -
 ADSL modem (Purple)">
        <String name="productDescription">ADSL modem</String>
        <String name="productCode">adslmodem</String>
        <String name="colour">Purple</String>
      </Object>
    </CustomCollectionEntry>
    <CustomCollectionEntry key="6541">
      <CollectionCode>productList</CollectionCode>
      <EntryCode>3gwifimodem</EntryCode>
      <Object objectId="6541" formId="870 displayName="3gwifi - 3G
 Wireless Modem (Purple)"">
        <String name="productDescription">3G Wireless Modem</String>
        <String name="productCode">3gwifi</String>
        <String name="colour">Purple</String>
      </Object>
    </CustomCollectionEntry>
  </CustomCollectionEntryList>
</CustomCollectionEntrySearchResults>
```

### Faults

`NoSuchItemException`

This fault is returned when:

- a collection code refers to a non-existent custom collection
- the `<Attribute>` element refers to a non-existent custom field

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.FindCollectionEntries
 collectionCode filename limit offset
```

***collectionCode***

**Purpose:** Specifies the custom collection to be searched.

***filename***

**Purpose:** Specifies the filename of the SearchCondition XML document that contains the content for the criteria argument.

***limit***

**Purpose:** Specifies the maximum number of results to return. moreResults="true" will be returned in result if more results exist beyond this limit. Must be a positive integer not exceeding 200.

***offset***

**Purpose:** Specifies the offset into the result set of where to begin returning results. Must be a non-negative integer.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage custom fields

Managing custom fields lets you create, update and delete entries from custom collections.

The following methods manage custom field entries:

- `addCollectionEntry()` creates a new collection entry for a custom collection.
- `updateCollectionEntry()` updates the custom fields for an existing custom collection entry.
- `deleteCollectionEntry()` deletes a custom collection entry.

### `addCollectionEntry()`

```
CustomCollectionEntry addCollectionEntry(
 NewCustomCollectionEntryRequest newEntryRequest)
```

This method creates a new collection entry for a custom collection.

#### Parameters

**newEntryRequest**

**Format:** NewCustomCollectionEntryRequest XML

**Purpose:** Defines a NewCustomCollectionEntryRequest XML document that specifies which custom collection the new entry will be added to, as well as the entry code name, and any custom fields to be defined. The layout of the custom fields is determined by the definition of the custom collection type in the Smile Configuration UI.

#### Example: Request a new custom collection entry

This example shows a NewCustomCollectionEntryRequest XML document to add a new entry to a custom collection.

```
<NewCustomCollectionEntryRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <CollectionCode>products</CollectionCode>
  <EntryCode>iphone6s</EntryCode>
  <Object>
    <String name="productCode">iPhone6S</String>
    <String name="productDesc">Apple iPhone 6S</String>
  </Object>
</NewCustomCollectionEntryRequest>
```

#### Results

This method returns a CustomCollectionEntry XML document describing the newly created custom collection entry. The entry includes a numeric key attribute that can be used to update or delete this entry at a later date.

The key is the identifier for the CustomCollectionEntry and objectId is the identifier for the custom fields within it. These values will not necessarily be the same.

Smile emits the `CustomCollectionEntryAdded` message when the task is complete.

### Example: A returned CustomCollectionEntry XML document

This example shows a CustomCollectionEntry XML document that contains an entry key of 4997.

```
<CustomCollectionEntry xmlns="http://xml.inomial.com/smile/2.xsd"
 key="4997">
  <CollectionCode>products</CollectionCode>
  <EntryCode>iphone6s</EntryCode>
  <Object objectId="4997" formId="1252" displayName="iphone6s - Apple
 iPhone 6S">
    <String name="productCode">iphone6s</String>
    <String name="productDesc">Apple iPhone 6S</String>
  </Object>
</CustomCollectionEntry>
```

### Faults

`NoSuchItemException`

This fault is returned when the custom collection in CollectionCode does not exist.

`NonUniqueFieldException`

This fault is returned when the EntryCode provided already exists.

`CustomObjectException`

This fault is returned when one or more custom fields inside the `<Object>` element are missing (in the case of a mandatory field), or fail to validate against a constraint that has been set for the content of that field.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddCollectionEntry filename
```

***filename***

**Purpose:** Specifies the filename of the NewCustomCollectionEntryRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### updateCollectionEntry()

```
void updateCollectionEntry(CustomCollectionEntry entry)
```

This method updates the existing custom collection entry identified by the key attribute of the entry provided.

Optionally it is also possible to rename the entry code of a custom collection entry, providing the new entry code is not already in use.

#### Parameters

**entry**

**Format:** CustomCollectionEntry XML

**Purpose:** Defines a CustomCollectionEntry XML document that specifies the details of the collection entry.

The key attribute of the top-level `<CustomCollectionEntry>` element in this argument is the sole identifier of which custom collection entry is to be updated.

The key of a custom collection is always returned by `addCollectionEntry()`, but you can also look up the key using `findCollectionEntries()` and `getCollectionEntryByEntryCode()`.

The `<CollectionCode>` element must contain the name of the custom collection that the entry belongs to. It is not possible to move a custom collection entry between different custom collections.

To rename the entry code of the custom collection entry, provide a different value for the `<EntryCode>` element. The `<Object>` element in this argument will replace the existing custom object associated with the custom collection entry being updated. You must include all the fields that you wish to retain post-update - with their new values if changing, or existing values if not to be changed.

You can use the return result of `getCollectionEntry()` as a basis for populating this argument.

#### Example: Request collection entry update

This example shows a CustomCollectionEntry XML document.

```
<CustomCollectionEntry xmlns="http://xml.inomial.com/smile/2.xsd"
 key="6544">
  <CollectionCode>AirFreightDestinations</CollectionCode>
  <EntryCode>AU_Melbourne</EntryCode>
  <Object>
    <String name="airportName">Tullamarine</String>
    <String name="airportIcaoCode">YMML</String>
    <String name="airportIataCode">MEL</String>
  </Object>
</CustomCollectionEntry>
```

#### Results

This method does not return any output.

Smile emits the `CustomCollectionEntryUpdated` message when the task is complete.

### Faults

`NoSuchItemException`

This fault is returned when a custom collection with the provided key does not exist.

`NonUniqueFieldException`

This fault is returned when an attempt to rename the entry code of a custom collection entry fails because an existing entry has the same name as the `<EntryCode>` element value.

`CustomObjectException`

This fault is returned when one or more custom fields inside the `<Object>` element are missing (in the case of a mandatory field), or fail to validate against a constraint that has been set for the content of that field.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateCustomCollectionEntry
 filename
```

***filename***

**Purpose:** Specifies the filename of the CustomCollectionEntry XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `deleteCollectionEntry()`

```
void deleteCollectionEntry(int key)
```

This method deletes a custom collection entry.

#### Parameters

**key**

**Format:** integer

**Purpose:** Specifies the custom collection entry to be deleted.

The key of a custom collection is always returned by `addCollectionEntry()`, but you can also look up the key using `findCollectionEntries()` and `getCollectionEntryByEntryCode()`.

#### Results

This method does not return any output.

Smile emits the `CustomCollectionEntryDeleted` message when the task is complete.

#### Faults

`NoSuchItemException`

This fault is returned when no custom collection entry exists with the given key.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.DeleteCustomCollectionEntry key
```

***key***

**Purpose:** Specifies the custom collection entry to be deleted.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Chapter

# 13

# Managing tickets

## CRM API overview

Managing tickets lets you manage workflows, which define the steps through which a process moves as it progresses from initiation to completion. Smile supports many types of workflows. For example, a workflow can manage helpdesk tickets or track the status of an order as it moves between departments.

Methods that manage tickets are located on the CRM endpoint. The CRM WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/crm?wsdl`

### Related reference

CRM endpoint on page 431
SmileTicket on page 479



inomial

## Retrieve ticket information

Retrieving ticket information lets you view details of a single or multiple tickets, details of a ticket template and the availability of appointment times.

The following methods manage tickets:

- `getTicket()` checks that a ticket number is valid.
- `getTickets()` retrieves all tickets for an account.
- `getTemplateTicketDescription()` retrieves the details of a template ticket.
- `getAppointmentAvailabilities()` searches for available operator appointment times.

### `getTicket()`

```
Ticket getTicket(string ticketNumber)
```

This method searches for a ticket by ticket number and returns information about the ticket. Use this method to check if a ticket number is valid.

#### Parameters

**ticketNumber**

**Format:** string

**Purpose:** Defines the number of the ticket to retrieve.

#### Results

This method returns a Ticket XML document that contains the valid ticket number and subject. If the ticket number is not valid, this method returns a fault.

There are no messages defined by this method.

#### Example: A returned Ticket XML document

This example shows a Ticket XML document with a valid ticket number.

```
<Ticket xmlns="http://xml.inomial.com/smile/2.xsd">
  <Number>184213</Number>
  <Subject>Close subscription 2142423645</Subject>
</Ticket>
```

#### Faults

`CrmException`

This fault is returned when the ticket cannot be returned or the caller does not have permission to access the ticket.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTicket ticketNumber
```

***ticketNumber***

**Purpose:** Specifies the number of the ticket to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getTickets()`

```
TicketList getTickets(string usn)
```

This method returns a list of all tickets for a specified account.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the account USN.

#### Results

This method returns a list of all tickets for a specified account USN.

There are no messages defined by this method.

#### Example: A returned TicketList XML document

This example shows a TicketList XML document when account USN 2142423223 is given as a parameter. The values in the nested `<Objects>` element, in the `<Properties>` element, display the name and value of custom fields, which are set for each type of ticket.

```
<TicketList xmlns="http://xml.inomial.com/smile/2.xsd" xmlns="http://
ws.inomial.com/smile.2">
  <Ticket>
    <Number>336</Number>
    <Subject>Test ticket</Subject>
    <TicketType key="382">Testing website ticket</TicketType>
    <TicketStatus key="1">New</TicketStatus>
    <AccountUSN>2142423215</AccountUSN>
    <SubscriptionUSN>2142423223</SubscriptionUSN>
    <Body>Test ticket</Body>
    <Created>2014-08-12T13:56:43.863+10:00</Created>
    <Opened>2014-08-12T13:56:43.953+10:00</Opened>
    <Properties>
      <Object>
        <String name="supportIssueType">Other</String>
      </Object>
    </Properties>
  </Ticket>
  <Ticket>
    <Number>335</Number>
    <Subject>EziDebit Refund</Subject>
    <TicketType key="397">Ezidebit refund</TicketType>
    <TicketStatus key="1">New</TicketStatus>
    <AccountUSN>2142423215</AccountUSN>
    <Created>2014-08-12T13:56:14.724+10:00</Created>
    <Opened>2014-08-12T13:56:14.857+10:00</Opened>
```

```
    <Properties>
      <Object>
        <String name="TransactionNumber">21579610</String>
        <String name="EziDebitReference">651398</String>
        <String name="FetchEmailAddress">joe@example.com</String>
        <Decimal name="RefundAmount">167.00</Decimal>
      </Object>
    </Properties>
  </Ticket>
</TicketList>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTickets usn
```

***usn***

**Purpose:** Specifies the account USN.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getTemplateTicketDescription()`

```
TemplateTicketDescription getTemplateTicketDescription(string
 templateTicketNumber)
```

This method retrieves the details of a template ticket. The returned document describes the template ticket's default values for the ticket, as well as the description of the ticket's custom form definition.

This can be used to discover the ticket's definition to dynamically render a form for a customer or operator, to fill in on an external website, or to synchronise configuration.

### Parameters

**templateTicketNumber**

**Format:** string

**Purpose:** Specifies the Smile template ticket number of which to retrieve details.

### Results

This method returns a TemplateTicketDescription XML document containing the details of the template ticket.

There are no messages defined by this method.

### Example: A returned TemplateTicketDescription XML document

This example shows a TemplateTicketDescription XML document for template ticket 127.

```
<TemplateTicketDescription xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <TemplateTicket>
    <Number>127</Number>
    <Subject>Refund Request</Subject>
    <TicketType key="381">Refund Request</TicketType>
    <TicketStatus key="1">New</TicketStatus>
    <Created>2014-05-03T17:41:23.833+10:00</Created>
    <Opened>2014-05-03T17:41:23.872+10:00</Opened>
    <Closed>2014-05-03T17:41:23.872+10:00</Closed>
    <Properties>
      <Object>
        <String name="ReceiptNumber">1239874213</String>
      </Object>
    </Properties>
  </TemplateTicket>
  <Form>
    <Attributes>
      <Attribute>
        <Required>false</Required>
        <Label>Refund Amount</Label>
        <Visibility>VisibleAndEditable</Visibility>
        <AttributeType>
          <AttributeType>469</AttributeType>
          <Name>RefundAmount</Name>
          <Label>Refund Amount</Label>
          <SmileObjectXMLType type="Decimal"/>
        </AttributeType>
      </Attribute>
      <Attribute>
        <Required>false</Required>
        <Label>Receipt Number</Label>
        <Visibility>VisibleAndEditable</Visibility>
        <AttributeType>
          <AttributeType>468</AttributeType>
          <Name>Receipt Number</Name>
          <Label>Receipt Number</Label>
          <SmileObjectXMLType type="String"/>
        </AttributeType>
      </Attribute>
      <Attribute>
        <Required>false</Required>
        <Label>Refund transaction number</Label>
        <Visibility>VisibleAndEditable</Visibility>
        <AttributeType>
          <AttributeType>472</AttributeType>
          <Name>TransactionNumber</Name>
          <Label>Refund transaction number</Label>
          <SmileObjectXMLType type="String"/>
        </AttributeType>
      </Attribute>
    </Attributes>
```

```
  </Form>
</TemplateTicketDescription>
```

### Faults

`NoSuchItemException`

This fault is returned when the requested template ticket number does not exist or does not refer to a template ticket.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTemplateTicketDescription
 templateTicketNumber
```

***templateTicketNumber***

**Purpose:** Specifies the Smile template ticket number of which to retrieve details.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getAppointmentAvailabilities()`

```
AppointmentAvailability getAppointmentAvailabilities(
 AppointmentAvailabilitiesRequest availabilitiesRequest)
```

This method returns list of appointment availabilities based on the request.

### Parameters

**availabilitiesRequest**

**Format:** AppointmentAvailabilitiesRequest XML

**Purpose:** Defines a new AppointmentAvailabilitiesRequest XML document that specifies the appointment availabilities searching criteria.

### Example: Search the availabilities of operators in a group

This example searches the availabilities of operators of a group.

```
<AppointmentAvailabilitiesRequest xmlns="http://xml.inomial.com/
smile/2.xsd">
  <TicketGroupId>322</TicketGroupId>
  <Start>2013-09-25T00:00:00+11:00</Start>
  <End>2013-09-25T23:59:59+11:00</End>
  <Duration>1800000</Duration>
  <TravelDuration>60000</TravelDuration>
</AppointmentAvailabilitiesRequest>
```

### Results

This method returns an AppointmentAvailability XML document which lists the search result.

### Faults

`InvalidRequestException`

This fault is returned when:

- the specified USN is not a member of the specified group
- neither USN or ticket group is provided
- appointment duration is not provided or not a multiple of 1 minute
- travel duration is negative or is not a multiple of 1 minute
- start time is not provided
- search end time is before the search start time

`NoSuchItemException`

This fault is returned when the specified USN is not an active subscription or group does not exist.

This error also occurs when there is no availability to return.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetAppointmentAvailabilities
 filename
```

***filename***

**Purpose:**Defines the filename of the AppointmentAvailabilitiesRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage tickets

Managing tickets lets you create and update tickets and workflows, run ticket actions and add ticket attachments.

The following methods manage tickets:

- `raiseTicket()` creates a ticket and starts a workflow.
- `runTicketAction()` retrieves information about a ticket after running a ticket action.
- `addTicketAttachment()` adds an attachment to a ticket.
- `updateTicket()` updates a tickets' subject, body text, custom fields or owning account or subscription details.

### `raiseTicket()`

```
Ticket raiseTicket(string templateTicketNumber, string usn,
 NewTicketRequest newTicketRequest)
```

This method creates a ticket and starts a workflow. This method also has the ability to schedule an operator appointment. The workflow to start and the ticket fields to complete are defined in the ticket template.

Before you use this method, configure the workflow and ticket template in Smile.

#### Parameters

**templateTicketNumber**

**Format:** string

**Purpose:** Specifies the template to use to create the ticket.

**usn**

**Format:** string

**Purpose:** Specifies the USN of the account or subscription to raise the ticket against.

**newTicketRequest**

**Format:** NewTicketRequest XML

**Purpose:** Defines a NewTicketRequest XML document that specifies the information to add to the ticket.

#### Example: Create a ticket

This example creates a ticket. The ticket fields to complete are defined in the ticket template.

```
<NewTicketRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <Properties>
    <Object>
      <String name="Plan">14.95</String>
      <Date name="AuthDate">2011-11-25</Date>
      <Date name="DateofBirth">1984-05-20</Date>
      <String name="DriversLicence">09123823</String>
```

```
      <Boolean name="PortExistingNumber">false</Boolean>
      <String name="Company">Example</String>
      <String name="ContactFamilyName">Smith</String>
      <String name="ContactGivenName">Joe</String>
      <String name="EmailAddress">joe@example.com</String>
      <String name="HomePhone">+61355501256</String>
      <String name="WorkPhone">0491570156</String>
      <String name="StreetName">Simpson</String>
      <String name="StreetNumber">60</String>
      <String name="Postcode">2000</String>
      <String name="Suburb">Sydney</String>
      <String name="FullNationalNumber">0491570156</String>
      <String name="State">NSW</String>
      <String name="StreetType">ESP</String>
    </Object>
  </Properties>
</NewTicketRequest>
```

### Example: Create a ticket to close a subscription

This example creates a ticket to close a subscription. The ticket fields to complete are defined in the ticket template.

```
<NewTicketRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <Properties>
    <Object>
      <Reference name="subscription">2142423645</Reference>
      <Date name="closeDate">2011-11-28</Date>
      <String name="closeReason">Customer is moving to the US.</
String>
    </Object>
  </Properties>
</NewTicketRequest>
```

### Example: Create an appointment

This example creates a 60 minute appointment with operator 2142422464, starting at midnight September 25, 2013.

```
<NewTicketRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <Properties</>
  <Subject>New Ticket Subject (customBody)</Subject>
  <Body/>
  <AppointmentRequest>
    <USN>2142422464</USN>
    <Start>2013-09-25T00:00:00+11:00</Start>
    <Duration>600000</Duration>
    <TravelDuration>0</TravelDuration>
  </AppointmentRequest>
</NewTicketRequest>
```

### Results

This method creates a ticket and returns a Ticket XML document with the ticket number for the new ticket. The method also starts the workflow defined in the ticket template.

There are no messages defined by this method.

### Example: A returned Ticket XML document

This example shows a Ticket XML document returned by the previous example. Note that the document contains the ticket number for the new ticket.

```
<Ticket xmlns="http://xml.inomial.com/smile/2.xsd">
  <Number>184213</Number>
  <Subject>Close subscription 2142423645</Subject>
</Ticket>
```

### Faults

`CrmException`

This fault is returned when the ticket cannot be created or the caller does not have permission to raise a ticket.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RaiseTicket templateTicketNumber
 usn filename
```

***templateTicketNumber***

**Purpose:** Specifies the template to use to create the ticket.

***usn***

**Purpose:** Specifies the USN of the account or subscription to raise the ticket against.

***filename***

**Purpose:** Defines the filename of the NewTicketRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### runTicketAction()

```
Ticket runTicketAction(string ticketNumber, int ticketAction)
```

This method returns information about a specified ticket after running a ticket action.

#### Parameters

**ticketNumber**

**Format:** string

**Purpose:** Specifies the ticket number.

**ticketAction**

**Format:** integer

**Purpose:** Specifies the ticket action ID. The ID is the Primary Key from TicketAction table.

#### Results

This method returns information about a given ticket after running a ticket action.

There are no messages defined by this method.

#### Example: A returned Ticket XML document

This example shows a Ticket XML document when ticket number 183 and 'Close ticket' ticket action are given as parameters.

```
<Ticket xmlns="http://xml.inomial.com/smile/2.xsd"
xmlns="http://ws.inomial.com/smile.2">
  <Number>183</Number>
  <Subject>Support call</Subject>
  <TicketType key="370">Support ticket</TicketType>
  <TicketStatus key="1">New</TicketStatus>
  <AccountUSN>2142420815</AccountUSN>
  <Created>2014-05-01T09:38:11.370+10:00</Created>
  <Opened>2014-05-01T09:39:02.704+10:00</Opened>
  <Properties>
    <Object>
      <String name="supportIssueType"></String>
    </Object>
  </Properties>
</Ticket>
```

#### Faults

`NoSuchItemException`

This fault is returned when the specified ticket number does not match a ticket.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.RunTicketAction ticketNumber
 ticketAction
```

***ticketNumber***

**Purpose:** Specifies the ticket number.

***ticketAction***

**Purpose:** Specifies the ticket action ID.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `addTicketAttachment()`

```
void addTicketAttachment(string ticketNumber, Attachment attachment)
```

This method adds an attachment to a ticket. Ticket attachments may be a URL or a directly attached file.

URLs are attached as links from the ticket and are not downloaded to Smile. Operators must have access to the URL provided for as long as the attachment is to be used.

#### Parameters

**ticketNumber**

**Format:** string

**Purpose:** Specifies the Smile ticket number to which the attachment is added.

**attachment**

**Format:** Attachment XML

**Purpose:** Defines an Attachment XML document that specifies the details of the attachment to add.

#### Example: Adding a URL attachment

This example shows how to add a URL attachment to a ticket.

```
<Attachment xmlns="http://xml.inomial.com/smile/2.xsd">
  <name>Signed Contract</name>
  <url>http://example.com/871bjiad8h/contract.pdf</url>
</Attachment>
```

#### Example: Uploading a file attachment

This example shows how to upload a file attachment to a ticket.

```
<Attachment xmlns="http://xml.inomial.com/smile/2.xsd">
  <name>Signed Contract</name>
  <mimeType>application/pdf</mimeType>
  <filename>contract.pdf</filename>
  <data><!-- bytes encoded as base64Binary --></data>
  <description>The contract signed by the customer</description>
</Attachment>
```

### Results

This method adds an attachment link to a Smile ticket.

This method does not return any output. There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when the provided ticket number does not exist or does not refer to a ticket.

`InvalidRequestException`

This fault is returned when the provided attachment document is not valid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddTicketAttachment ticketNumber
 filename
```

***ticketNumber***

**Purpose:** Specifies the Smile ticket number to which the attachment is added.

***filename***

**Purpose:** Defines the filename of the Attachment XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `updateTicket()`

```
void updateTicket(string ticketNumber, TicketUpdate ticketUpdate)
```

This method updates an existing tickets' subject, body text, custom fields or owning account or subscription. It can be used by external systems which need to populate data into tickets.

### Parameters

**ticketNumber**

**Format:** string

**Purpose:** Specifies the Smile ticket number to update.

**ticketUpdate**

**Format:** TicketUpdate XML

**Purpose:** Defines a TicketUpdate XML document that specifies the ticket details to update.

### Example: Parameter title

This example shows a TicketUpdate XML document with updates to the ticket's subject and body fields, as well as setting the ticket to be owned by account/subscription 2142420625.

In the case where the USN refers to a subscription, the ticket's account is automatically updated to be the subscription's account. To clear the ticket's account or subscription, set clear to true on the USN tag.

```
<TicketUpdate xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subject>Updated ticket subject</Subject>
  <USN clear="false">2142420625</USN>
  <Body>New ticket body text</Body>
  <Properties>
    <Object>
      <String name="ModemSerialNumber">aabbccddeeff</String>
    </Object>
  </Properties>
</TicketUpdate>
```

### Results

This method updates an existing tickets' subject, body text, custom fields or owning account or subscription as given in the TicketUpdate XML document.

This method does not return any output. There are no messages defined by this method.

### Faults

`NoSuchItemException`

This fault is returned when the provided ticket number does not exist or does not refer to a ticket.

`InvalidRequestException`

This fault is returned when the update request is not valid.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateTicket ticketNumber
 filename
```

***ticketNumber***

**Purpose:** Specifies the Smile ticket number to update.

***filename***

**Purpose:** Defines the filename of the TicketUpdate XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Chapter

# 14

# Listening for financial transactions

## Open item accounts

Smile is an open item accounts receivable system. A basic understanding of how open item accounting works will help you understand how transactions and accounts receivable work in Smile.

Open item accounting is different to balance-forward accounting. Balance-forward accounting treats the balance of each account as a pool of money. When a customer makes a payment, the account balance decreases; when you issue an invoice to a customer, the account balance increases. However, in a balance-forward accounts system you cannot assign a payment to a specific invoice.

In contrast, an open item accounts system allocates every payment to an open item (a transaction with an outstanding balance). Transactions are allocated to open items until the item has a balance of zero and is closed.

In an open item accounts system, you can specify the invoices that a payment is allocated to. For example, a customer may not want to pay an invoice in dispute, but is happy to pay subsequent invoices. Using an open item accounts system, you can allocate the customer's payments to the later invoices and leave the disputed invoice unpaid.

## Transaction overview

A transaction is a financial event that affects an account. Transactions are the basis of Smile's accounts receivable system. Smile accounts have an account history that lists every transaction raised against each account throughout the account's lifecycle.

### Allocated transactions

Because Smile is an open item accounts system, each transaction can be allocated to another transaction. For example, you can allocate a surcharge reversal to a surcharge, and allocate an invoice to a receipt. A receipt is an acknowledgement of a completed payment.

An allocated transaction does not have to pay an open item in full, and a transaction can have more than one transaction allocated to it. For example, a receipt can be allocated to both an invoice and a surcharge, as shown in the following diagram.

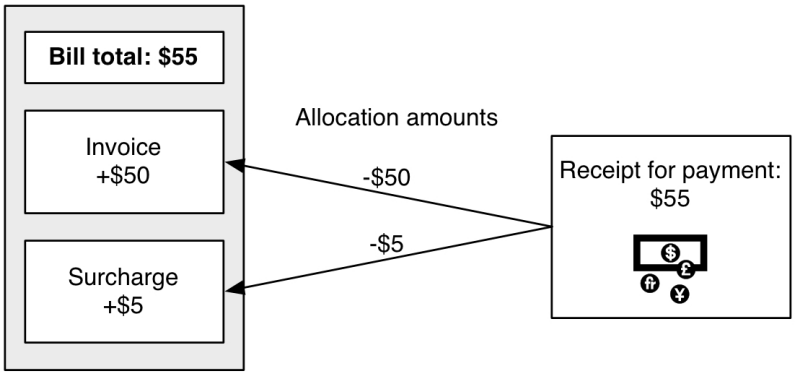



**Figure 10: Receipt allocated to multiple transactions**

Smile automatically allocates transactions to each other, with some exceptions. For example, Smile automatically allocates a credit card payment to an invoice, but an operator manually allocates a cash payment to an invoice.

### Debit and credit transactions

In Smile, credit transactions have negative values because they decrease the account balance, that is, decrease the amount the customer owes. Debit transactions have positive values because they increase the account balance, that is, increase the amount the customer owes.

The following diagram shows you how the credit and debit transactions in the previous example appear on an account balance. In this example, the debit transactions are shaded red and the credit transaction is shaded green.

| Item | Amount | Account bal |
|---|---|---|
| Invoice | +$50 | +$50 |
| Surcharge | +$5 | +$55 |
| Receipt | -$55 | $0 |

Bill

**Figure 11: Debit and credit transactions**

### Itemised transactions

An itemised transaction lists the charges within the transaction. For example, an invoice contains a list of the invoice items that make up the invoice balance. An itemised credit lists the credit items within a credit note, for example, a $100 itemised credit may list an $80 refund and a $20 goodwill credit.

## Transaction types

A transaction type is a type of financial event that affects an account. You can perform the following transaction types in Smile:

| | |
|---|---|
| **Quote** | A formal statement of the cost to provide services and equipment to a customer. An open quote does not affect a customer's account balance. When a quote is closed, Smile adds the value of the quote to the account balance. |
| **Invoice** | A list of itemised charges raised against a subscription or account over a given period of time. |
| **Surcharge** | An extra fee added to another fee or charge, for example, a fee for a paper bill or a credit card payment. |
| **Receipt** | An acknowledgement of a completed payment. |
| **Credit note** | An adjustment that decreases the account balance, that is, decreases the amount the customer owes. Credit notes have negative values and appear in the account history and sales reports. You can use credit notes to compensate customers for an inconvenience. |
| **Debit note** | An adjustment that increases the account balance, that is, increases the amount the customer owes. Debit notes have positive values and appear in the account history and sales reports. You can use debit notes to correct invoices that have undercharged a customer. |
| **Itemised credit** | A credit note that contains an itemised list of the charges within the credit. Itemised credits provide more information about why you issued the credit. For example, a $100 itemised credit may contain an $80 refund and a $20 goodwill credit, which you can track separately in reports. Itemised credits appear in the account history, and the individual items appear in sales reports. |
| **Itemised debit** | A debit note that contains an itemised list of charges that make up the debit. Itemised debits provide more information about why you issued the debit. Itemised debits appear in the account history, and the individual items appear in sales reports. |
| **Recipient created tax invoice** | An invoice, created by and addressed to a company, from a creditor requesting payment. Smile generates recipient created tax invoices (RCTIs) on behalf of creditors to manage commission payments. RCTIs include GST where applicable. |
| **Credit transferred credit** | A transaction that results when a receipt is allocated to an invoice from a different account. A credit transferred credit appears in the account history of the account the receipt is allocated to, that is, the account with the invoice, and decreases the account balance. |

**Credit transferred debit** A transaction that results when a receipt is allocated to an invoice from a different account. A credit transferred debit appears in the account history of the account that the receipt is allocated from, that is, the account the receipt was originally raised against, and increases the account balance.

**Transferred credit** A transaction that results from a transfer of funds between accounts. A transferred credit appears in the account history of the account whose account balance decreases, that is, decreases the amount the customer owes.

**Transferred debit** A transaction that results from a transfer of funds between accounts. A transferred debit appears in the account history of the account whose account balance increases, that is, increases the amount the customer owes. Refer to the following table for more information about how transferred credits and debits appear in the account history.

| Transfer from account A to B | Transaction type in A's Account History | Transaction type in B's Account History |
|---|---|---|
| credit (A sends money to B) | Transferred debit | Transferred credit |
| debit (A sends a debit to B) | Transferred credit | Transferred debit |

## Reversals

A reversal is a transaction that negates another transaction. Reversals are used to correct mistakes, for example, if a cheque bounces after you enter a payment into Smile, you can reverse the receipt and remove the credit from the account balance.

Reversals are listed in the account history and sometimes appear in statements and sales reports, as follows:

- If the original transaction and the reversal both occur within the dates of the statement range, they do not appear on the statement and sales report.
- If one transaction occurs within the dates of the statement range, but the corresponding transaction occurs outside the statement range, the transaction in range appears on the statement and sales report.

### Example: Reversals and statement ranges

The following diagram shows you an example of when a reversal appears on a statement. This example has two transactions, a surcharge that occurs on 30 April and a surcharge reversal that occurs on 2 May.

- If the statement range is 15 April-14 May, the surcharge and reversal do not appear on the statement.
- If the statement range is 1 May-31 May, the surcharge does not appear, but the reversal does appear on the statement.

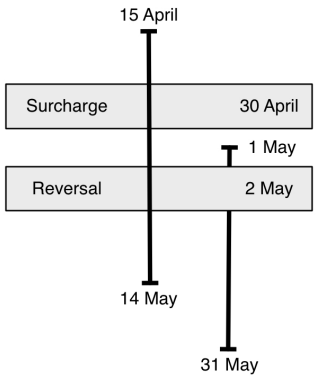



**Figure 12: Statement ranges and reversals**

## Transaction lifecycle

The transaction lifecycle describes the actions you can perform on a transaction before the debtor close date. Once the debtor close date for a transaction has passed, the transaction is locked and you cannot make any changes to it.

The debtor closed date lags behind real time to give you a window of time to correct mistakes before the transaction is locked, and to ensure long-term report stability. Your accounts team moves the debtor closed date forward each month or at an interval that suits your business processes.

The stages in the transaction lifecycle are:

| | |
|---|---|
| **Closed** | Smile closes the transaction and adds the transaction value to the account balance.<br>Transactions may be open for some time before they are closed, for example, quotes or credit card payments pending bank approval. Other transactions are closed immediately, for example, invoices, receipts for cash payments and adjustments. |
| **Allocated** | Smile allocates the transaction to another transaction. If necessary, you can also deallocate transactions. |
| **Reversed** | Smile raises a reversal transaction against an existing transaction, then allocates the reversal to the original transaction. Both transactions appear in the account history.<br>Use a reversal when you cannot delete a transaction or when you want a record of the reversal to appear in reports. |
| **Reopened** | Smile reopens the transaction and any items in the transaction, and removes the transaction value from the account balance. You can now make financial changes to the transaction.<br>After you close a reopened transaction, Smile adds the new transaction value to the account balance and updates the transaction. |
| **Updated** | Smile updates the transaction's due date or comments, but does not reopen the transaction. The transaction value remains unchanged and on the account balance. |
| **Deleted** | Smile deletes the transaction, removes the transaction value from the account balance and removes the transaction from the account history. Smile records the delete action in the logs to maintain an audit trail of the event.<br>You cannot delete a transaction that will affect the integrity of other Smile data. For example, you cannot delete the receipt for a credit card payment because Smile keeps a record of the payment approval. |

Smile emits a SmileTransaction message each time a transaction changes. The following diagram shows the messages that Smile emits at each stage of the transaction lifecycle:

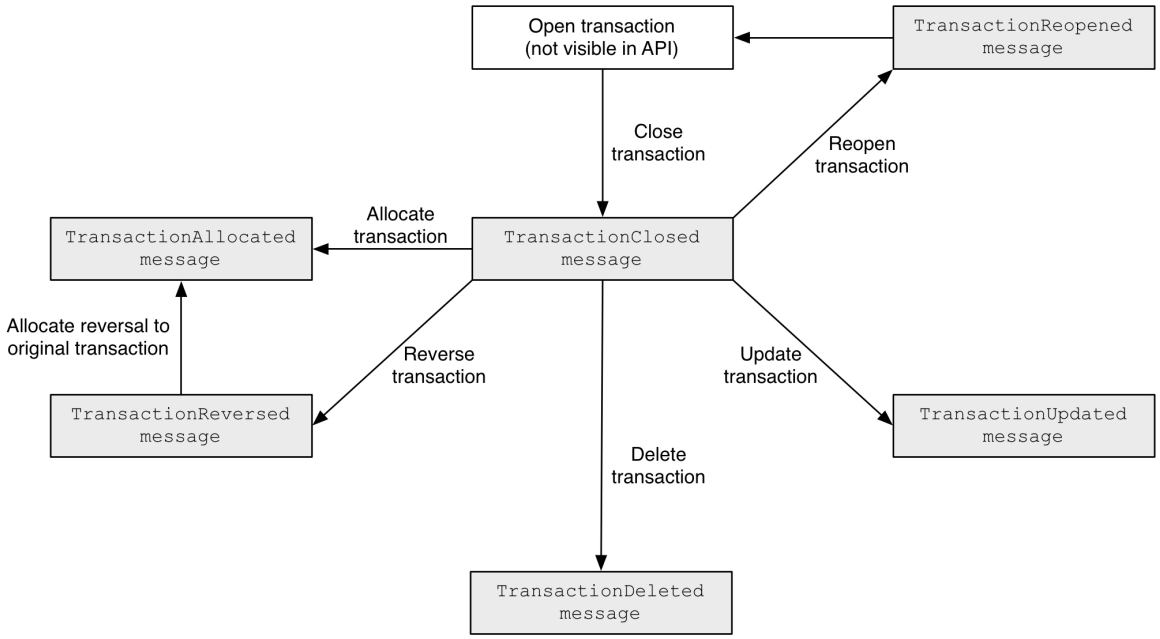



**Figure 13: The transaction lifecycle**

### Related reference

SmileTransaction on page 517

## Synchronise transaction information

Each time Smile closes or changes a transaction, it emits a message on the SmileTransaction. Listening for these transaction messages lets you synchronise the transaction events in Smile with your general ledger accounts. For example, your accounts system can listen for the messages raised when Smile creates an invoice, or your reports system can listen for transactions to generate real-time reports.

Smile emits SmileTransaction messages when:

- `TransactionAllocated`—a transaction is allocated to another transaction.
- `TransactionClosed`—a transaction is closed and the transaction value is added to the account balance.
- `TransactionDeleted`—a transaction is deleted and the transaction value is removed from the account balance.
- `TransactionReopened`—a transaction and any items it contains are reopened and the transaction value is removed from the account balance.
- `TransactionReversed`—a new transaction is raised to reverse an existing transaction.
- `TransactionUpdated`—a reopened transaction is closed or the due date or comment is changed.

### Related reference

SmileTransaction on page 517

# Chapter 15

# Performing other Smile tasks

## Endpoint overview

The following SOAP endpoints let you perform a subset of tasks in an area of functionality:

- **authenticate**—authenticates credentials for Smile subscriptions
- **accounts receivable**—retrieves item descriptions
- **CAT**—retrieves CAT user password details
- **provisioning**—retrieves and updates the provision status of provision requests
- **recurring charge**—retrieves, updates and adds recurring charges
- **service**—retrieves service information and qualifies a service
- **pools**—retrieves and updates number pool reservations

You can access the full range of tasks for these areas of functionality in Smile.

## Authenticate credentials

Smile can authenticate subscription credentials that are passed to it by another application. For example, if a customer logs in to a web application, Smile can authenticate the customer's username and password against the username and password of their Smile subscription.

You cannot use the authentication endpoint to log in to Smile.

Methods that authenticate credentials are located on the authentication endpoint. The authentication WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/authentication?wsdl`

The `authenticate()` method authenticates a username and password.

### Related reference

Authentication endpoint on page 426

### `authenticate()`

```
AuthenticationResponse authenticate(AuthenticationRequest
 authenticationRequest)
```

This method authenticates credentials for Smile subscriptions and returns the subscription's USN. Use this method when you need to validate customer credentials.

You cannot use this method to log in to Smile.

#### Parameters

**authenticationRequest**

**Format:** AuthenticationRequest XML

**Purpose:** Defines an AuthenticationRequest XML document that specifies the credentials to authenticate.

The parameters in this document are determined by the authentication request type. The default authentication request type is 1 and validates a username and password. Contact Inomial to define additional authentication request types.

The `<serviceType>` element has the following key attributes:

| | |
|---|---|
| **1** | Smile |
| **2** | CAT |

For additional service types contact Inomial.

#### Example: Authenticate a username and password

This example authenticates a username and password. The authentication request type is 1.

```
<AuthenticationRequest xmlns="http://xml.inomial.com/smile/2.xsd">
  <AuthenticationRequestType>1</AuthenticationRequestType>
  <Parameters>
```

```
    <Object>
      <String name="username">joe@example.com</String>
      <String name="password">password</String>
      <Integer name="serviceType">2<Integer>
    </Object>
  </Parameters>
</AuthenticationRequest>
```

### Results

This method returns an AuthenticationResponse XML document that contains the authentication results. If the authentication is successful, the document also contains the subscription's USN.

There are no messages defined by this method.

### Example: A returned AuthenticationResponse document

This example shows a successful authentication response. Note the subscription's USN is included in the response.

```
<AuthenticationResponse xmlns="http://xml.inomial.com/smile/2.xsd">
  <Success>true</Success>
  <USN>1923102312</USN>
</AuthenticationResponse>
```

### Faults

`AuthenticationException`

This fault is returned when the request is invalid or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.Authenticate filename
```

***filename***

**Purpose:** Defines the filename of the AuthenticationRequest XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

Related concepts

Voucher use cases on page 313
Use self-serve prepaid on page 322

## Manage accounts receivable

Methods that manage accounts receivable are located on the accounts receivable endpoint. The accounts receivable WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/accountsreceivable?wsdl`

The following methods retrieve accounts receivable information:

- `getItemDescription()` returns details of an item description for the given company and item code.
- `getItemDescriptionById()` returns details of an item description.

Related reference

Accounts receivable endpoint on page 425

### `getItemDescription()`

```
LineItemDescription getItemDescription(int company, string itemCode)
```

This method returns details of an item description for the given company and item code.

#### Parameters

**company**

**Format:** integer

**Purpose:** Specifies the identification number of the company for which to retrieve the item description.

**itemCode**

**Format:** string

**Purpose:** Specifies the identification code of the item description.

#### Results

This method returns a LineItemDescription XML document that contains the properties on the item description.

There are no messages defined by this method.

#### Example: A returned LineItemDescription XML document

This example shows a LineItemDescription XML document.

```
<LineItemDescription xmlns="http://xml.inomial.com/smile/2.xsd">
  <Company>1</Company>
  <ItemCode>misc1234</ItemCode>
  <Name>Example Company Pty Ltd: Professional services</Name>
  <UnitSize>1</UnitSize>
  <ChargeGst>true</ChargeGst>
  <Rate formatted="$0.50" currency="AUD">0.50000</Rate>
  <ReportingCategory>
    <Company>1</Company>
```

```
    <ReportingCode>exampleps</ReportingCode>
    <Name>Example Company Pty Ltd: Professional services division</
Name>
    <Id>10</Id>
  </ReportingCategory>
  <Currency>AUD</Currency>
</LineItemDescription>
```

### Faults

`NoSuchItemException`

This fault is returned when the `company` and `itemCode` does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetItemDescription company
 itemCode
```

***company***

**Purpose:** Specifies the identification number of the company for which to retrieve the item description.

***itemCode***

**Purpose:** Specifies the identification code of the item description.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getItemDescriptionById()`

```
LineItemDescription getItemDescriptionById(string itemDescriptionId)
```

This method returns details of an item description.

### Parameters

**itemDescriptionId**

**Format:** string

**Purpose:** Specifies identification code for the item description.

### Results

This method returns a LineItemDescription XML document that contains the properties of the item description.

There are no messages defined by this method.

### Example: A returned LineItemDescription XML document

This example shows a LineItemDescription XML document.

```
<LineItemDescription  xmlns="http://xml.inomial.com/smile/2.xsd">
  <Company>1</Company>
```

```
  <ItemCode>misc1234</ItemCode>
  <Name>Example Company Pty Ltd: Professional services</Name>
  <UnitSize>1</UnitSize>
  <ChargeGst>true</ChargeGst>
  <Rate formatted="$0.50" currency="AUD">0.50000</Rate>
  <ReportingCategory>
    <Company>1</Company>
    <ReportingCode>exampleps</ReportingCode>
    <Name>Example Company Pty Ltd: Professional services division</
Name>
    <Id>10</Id>
  </ReportingCategory>
  <Currency>AUD</Currency>
</LineItemDescription>
```

### Faults

`NoSuchItemException`

This fault is returned when the `itemDescriptionId` does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetItemDescriptionById
 itemDescriptionId
```

***itemDescriptionId***

**Purpose:** Specifies identification code for the item description.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage CAT

Methods that manage CAT information are located on the CAT endpoint. The CAT WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/cat?wsdl`

The `triggerCatForgotPasswordEmail()` method retrieves and emails CAT user password details.

Related reference

CAT endpoint on page 427

### triggerCatForgotPasswordEmail()

```
void triggerCatForgotPasswordEmail(string catUsername)
```

This method generates an email containing the CAT login password that is sent to the CAT users stored email address. This email needs to be configured in Smile otherwise an exception will be generated.

#### Parameters

**catUsername**

**Format:** string

**Purpose:** Specifies the CAT username of which to retrieve the password.

#### Results

This method generates an email containing the user's CAT login password. The email is sent to the user's email address stored in Smile.

There are no messages defined by this method.

#### Faults

`CatException`

This fault is returned when the specified username cannot be found or user does not have permission to call the method.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.TriggerCatForgotPasswordEmail
 catLoginUsername
```

***catLoginUsername***

**Purpose:** Specifies the CAT username of which to retrieve the password.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage provisioning

Managing provisioning allows you to retrieve and update the status of a provision request.

Methods that manage provisioning are located on the provisioning endpoint. The provisioning WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/provisioning?wsdl`

The following methods manage provisioning requests:

- `getIncompleteProvisionRequests` retrieves a list of currently incomplete provisioning requests that have been lodged for a subscription.
- `provisioningUpdate()` updates the provision status of a provision request with a log message

### Related reference

Provisioning endpoint on page 453

### getIncompleteProvisionRequests()

```
ProvisionRequests getIncompleteProvisionRequests(string usn)
```

This method retrieves a list of currently incomplete provisioning requests that have been lodged for a subscription.

#### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the USN of the subscription for which to retrieve incomplete provision requests.

#### Results

This method returns a ProvisionRequests XML document that contains zero or more incomplete provisioning requests for a subscription.

An incomplete provisioning request is defined as one that has a status other than "Complete".

There are no messages defined by this method.

#### Example: A returned ProvisionRequest XML document

This example shows a ProvisionRequests XML document for USN 2142421607.

```
<ProvisionRequests xmlns="http://xml.inomial.com/smile/2.xsd">
 <ProvisionRequest>
   <provisionRequestId>679</provisionRequestId>
   <company key="1">Example Telecom</company>
   <USN>2142421607</USN>
   <requestType>RequestProvision</requestType>
   <status>Transferring</status>
   <created>2015-06-11T19:38:16.489+10:00</created>
   <actionTime>2015-06-11T19:38:16.489+10:00</actionTime>
```

```
    <attemptLimit>1</attemptLimit>
    <result>Delayed</result>
    <ProvisionRequestLogs>
      <ProvisionRequestLog>
        <message>Provisioning request dispatched to external
  provider</message>
        <logged>2015-06-12T16:45:01.026+10:00</logged>
        <loggedBy>2142420732</loggedBy>
      </ProvisionRequestLog>
      <ProvisionRequestLog>
        <message>External provider indicated more time needed to
  complete provisioning request</message>
        <logged>2015-06-14T09:52:37+10:00</logged>
        <loggedBy>2142420732</loggedBy>
      </ProvisionRequestLog>
    </ProvisionRequestLogs>
  </ProvisionRequest>
</ProvisionRequests>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not correspond to an existing subscription.

`InvalidRequestException`

This fault is returned when:

- a USN is not specified
- the USN specified refers to an account, instead of a subscription

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetIncompleteProvisionRequests
 usn
```

***usn***

**Purpose:** Specifies the USN of the subscription for which to retrieve incomplete provision requests.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### provisioningUpdate()

```
ProvisioningResponse provisioningUpdate(int provisionRequestId,
 ProvisionRequestStatus status, string logMessage)
```

This method updates the provision status of a given provision request with a log message.

#### Parameters

**provisionRequestId**

**Format:** integer

**Purpose:** Specifies the provision request ID.

**status**

**Format:** ProvisionRequestStatus XML

**Purpose:** (Optional) Defines a ProvisionRequestStatus XML document that specifies the new status for the provision request.

**logMessage**

**Format:** string

**Purpose:** (Optional) Specifies a message that will be logged for the provision request.

**Note:** If both optional parameters are not specified, the method will return the details of the provision request.

#### Example: A ProvisionRequestStatus XML document

This example shows a ProvisionRequestStatus XML document with a status of TempFail.

The `<ProvisionRequestStatus>` element has one of the following values:

| | |
|---|---|
| **TempFail** | Indicates that a failure has occurred, but might be successful if retried. |
| **PermFail** | Indicates that a failure has occurred, but will be removed if retried. |
| **Delayed** | Indicates that the provision is still in progress. |
| **Provisioned** | Indicates that the provision is complete. |

```
<ProvisionRequestStatus xmlns="http://xml.inomial.com/
smile/2.xsd">TempFail</ProvisionRequestStatus>
```

#### Results

This method returns a ProvisioningResponse XML document that contains details of the provision request after update.

There are no messages defined by this method.

### Example: A returned ProvisioningResponse XML document

This example shows a ProvisioningResponse XML document for provision request ID 1018.

```
<ProvisioningResponse xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <provisionRequestId>1018</provisionRequestId>
  <company>1</company>
  <USN>2142425467</USN>
  <requestType>RequestProvision</requestType>
  <status>Complete</status>
  <created>2015-01-20T11:23:59.950+11:00</created>
  <actionTime>2015-01-20T11:24:00.106+11:00</actionTime>
  <attemptLimit>1</attemptLimit>
  <result>PermFail</result>
</ProvisioningResponse>
```

### Faults

`InvalidRequestException`

This fault is returned when no provision request ID is given as a parameter or the status is unknown.

`ProvisioningException`

This fault is returned when Smile is unable to log a provision request with logMessage or update the provision request.

`NoSuchItemException`

This fault is returned when Smile is unable to find the provision request with the given provision request ID.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.ProvisioningUpdate
 provisionRequestId [filename] [logMessage]
```

***provisionRequestId***

**Purpose:** Specifies the provision request ID.

***filename***

**Purpose:** (Optional) Defines the filename of the ProvisionRequestStatus XML document.

***logMessage***

**Purpose:** (Optional) Specifies a message that will be logged for the provision request.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage recurring charges

Methods that manage recurring charges are located on the recurring charge endpoint. The payment WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/recurringcharge?wsdl`

The following methods manage payments:

- `getSimpleRecurringCharge()` retrieves a recurring charge with a specified recurring charge ID.
- `getSimpleRecurringCharges()` retrieves a list of recurring charges for an account or subscription.
- `updateSimpleRecurringCharges()` updates recurring charges for an account or subscription.
- `addSimpleRecurringCharges()` adds recurring charges to an account or subscription.

Related reference

Recurring charge endpoint on page 454

### `getSimpleRecurringCharge()`

```
SimpleRecurringCharge getSimpleRecurringCharge(string
 userAutoChargeId)
```

This method returns a recurring charge with the specified recurring charge ID.

#### Parameters

**userAutoChargeId**

**Format:** string

**Purpose:** Specifies the ID of the recurring charge to be returned.

#### Results

This method returns a recurring charge with the specified recurring charge ID.

There are no messages defined by this method.

#### Example: A returned SimpleRecurringCharge XML document

This example shows a SimpleRecurringCharge XML document containing a recurring charge that has UserAutoCharge ID ‘476’ when it is given as a parameter.

```
<SimpleRecurringCharge xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <USN>2142420989</USN>
  <SimpleRecurringCharge>
    <RecurringCharge id="476">
      <ChargeQuantity>1</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
```

```
      <MinProRataDays>0</MinProRataDays>
      <LineItemDescription>
        <ItemCode>hfp</ItemCode>
        <Name>Handsfree wireless telephone handset</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate formatted="$75.00" currency="AUD">75.00000</Rate>
        <DisplayRate>$82.50</DisplayRate>
      </LineItemDescription>
      <ValidFrom>2014-09-10+10:00</ValidFrom>
   </RecurringCharge>
   <Rate formatted="$75.00" currency="AUD">75.00000</Rate>
   <DisplayRate>$82.50</DisplayRate>
  </SimpleRecurringCharge>
</SimpleRecurringCharge>
```

### Faults

`NoSuchItemException`

This fault is returned when the `<userAutoChargeId>` element does not match a recurring charge, or the specified `<LineItemDescription>` element in a recurring charge does not match an item description.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSimpleRecurringCharge
 userAutoChargeId
```

***userAutoChargeId***

**Purpose:** Specifies the ID of the recurring charge to be returned.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getSimpleRecurringCharges()`

```
SimpleRecurringCharges getSimpleRecurringCharges(string usn)
```

This method returns a list of recurring charges for a specified account or subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the account or subscription USN.

### Results

This method returns a list of recurring charges for a specified account or subscription USN.

There are no messages defined by this method.

### Example: A returned SimpleRecurringCharges XML document

This example shows a SimpleRecurringCharges XML document containing recurring charges that belongs to account USN 2142420989, which is given as a parameter.

```
<SimpleRecurringCharges xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <USN>2142420989</USN>
  <SimpleRecurringCharge>
    <RecurringCharge id="476">
      <ChargeQuantity>1</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
      <MinProRataDays>0</MinProRataDays>
      <LineItemDescription>
        <ItemCode>hfp</ItemCode>
        <Name>Handsfree wireless telephone handset</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate formatted="$75.00" currency="AUD">75.00000</Rate>
        <DisplayRate>$82.50</DisplayRate>
      </LineItemDescription>
      <ValidFrom>2014-09-10+10:00</ValidFrom>
    </RecurringCharge>
    <Rate formatted="$75.00" currency="AUD">75.00000</Rate>
    <DisplayRate>$82.50</DisplayRate>
  </SimpleRecurringCharge>
  <SimpleRecurringCharge>
    <RecurringCharge id="429">
      <ChargeQuantity>1</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
      <MinProRataDays>0</MinProRataDays>
      <LineItemDescription>
        <ItemCode>consult</ItemCode>
        <Name>Consulting and support services</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate formatted="$90.91" currency="AUD">90.90909</Rate>
        <DisplayRate>$100/hr</DisplayRate>
      </LineItemDescription>
      <ValidFrom>2014-09-01+10:00</ValidFrom>
    </RecurringCharge>
    <Rate formatted="$90.91" currency="AUD">90.90909</Rate>
    <DisplayRate>$100/hr</DisplayRate>
  </SimpleRecurringCharge>
</SimpleRecurringCharges>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription, or the specified `<LineItemDescription>` in a recurring charge does not match an item description.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetSimpleRecurringCharges usn
```

***usn***

**Purpose:** Specifies the account or subscription USN.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `updateSimpleRecurringCharges()`

```
void updateSimpleRecurringCharges(string usn,
 NewSimpleRecurringCharges recurringCharges)
```

This method is used for updating recurring charges of a specified account or subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the account or subscription USN.

**recurringCharges**

**Format:** NewSimpleRecurringCharges XML

**Purpose:** Defines a NewSimpleRecurringCharges XML document that specifies the values of the recurring charges to be updated.

### Example: A NewSimpleRecurringCharges XMI document

This example shows a NewSimpleRecurringCharges XML document to update recurring charge ID '515' with new values for display rate and frequency type.

```
<NewSimpleRecurringCharges xmlns="http://xml.inomial.com/
smile/2.xsd">
  <SimpleRecurringCharge>
    <RecurringCharge id="515">
      <Frequency type="Days">30</Frequency>
    </RecurringCharge>
    <DisplayRate>$150/hr</DisplayRate>
  </SimpleRecurringCharge>
</NewSimpleRecurringCharges>
```

### Results

No results returned for this update method.

### Faults

`RecurringChargeException`

This fault is returned when one of the following situations occur:

- A requested recurring charge belongs to another account or subscription, but not to the specified USN
- Unrecognised pro-rata action is provided
- Unrecognised frequency type is provided
- Unrecognised charging mode is provided

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription.

`InvalidRequestException`

This fault is returned when a requested recurring charge has already been cancelled and cannot be updated.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.UpdateSimpleRecurringCharges usn
 filename
```

***usn***

**Purpose:** Specifies the account or subscription USN.

***filename***

**Purpose:** Defines the filename of the NewSimpleRecurringCharges XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `addSimpleRecurringCharges()`

```
SimpleRecurringCharges addSimpleRecurringCharges(string usn,
 NewSimpleRecurringCharges recurringCharges)
```

This method is used for adding recurring charges to a specified account or subscription.

### Parameters

**usn**

**Format:** string

**Purpose:** Specifies the account or subscription USN.

**recurringCharges**

**Format:** NewSimpleRecurringCharges XML

**Purpose:** Defines a NewSimpleRecurringCharges XML document that specifies the values of recurring charges to be updated.

### Example: A NewSimpleRecurringCharges XML document

This example shows a NewSimpleRecurringCharges XML document to add a new recurring charge to the account or subscription USN specified.

```
<NewSimpleRecurringCharges xmlns="http://xml.inomial.com/
smile/2.xsd">
  <SimpleRecurringCharge>
    <RecurringCharge>
      <ChargeQuantity>2</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
      <MinProRataDays>0</MinProRataDays>
      <LineItemDescription>
        <ItemCode>consult</ItemCode>
        <Name>Consulting and support services</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate formatted="$200.00" currency="AUD">200</Rate>
        <DisplayRate>$100/hr</DisplayRate>
      </LineItemDescription>
      <ValidFrom>2014-09-01+10:00</ValidFrom>
    </RecurringCharge>
    <Rate formatted="$200.00" currency="AUD">200</Rate>
    <DisplayRate>$100/hr</DisplayRate>
  </SimpleRecurringCharge>
</NewSimpleRecurringCharges>
```

### Results

This method returns a list of new recurring charges added to the account or subscription USN specified.

There are no messages defined by this method.

### Example: A returned SimpleRecurringCharges XML document

This example shows a SimpleRecurringCharges XML document containing a new recurring charge that has been added to account USN 2142423447, which is given as a parameter.

```
<SimpleRecurringCharges xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <USN>2142423447</USN>
  <SimpleRecurringCharge>
    <RecurringCharge id="518">
      <ChargeQuantity>2</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
      <MinProRataDays>0</MinProRataDays>
      <LineItemDescription>
        <ItemCode>consult</ItemCode>
        <Name>Consulting and support services</Name>
```

```
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate formatted="$200.00" currency="AUD">200</Rate>
        <DisplayRate>$100/hr</DisplayRate>
      </LineItemDescription>
      <ValidFrom>2014-09-01+10:00</ValidFrom>
    </RecurringCharge>
    <Rate formatted="$200.00" currency="AUD">200</Rate>
    <DisplayRate>$100/hr</DisplayRate>
  </SimpleRecurringCharge>
</SimpleRecurringCharges>
```

### Faults

`NoSuchItemException`

This fault is returned when the specified USN does not match an account or subscription, or the specified `<LineItemDescription>` in a recurring charge does not match an item description.

`RecurringChargeException`

This fault is returned when one of the following situations occur:

- An unrecognised pro-rata action is provided
- An unrecognised frequency type is provided
- An unrecognised charging mode is provided

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.AddSimpleRecurringCharges usn filename
```

***usn***

**Purpose:** Specifies the account or subscription USN.

***filename***

**Purpose:** Defines the filename of the NewSimpleRecurringCharges XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage services

Managing services lets you view service descriptions, retrieve information about tariffs and check for service and feature availability with your upstream provider.

Methods that manage services are located on the service endpoint. The service WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/service?wsdl`

The following methods manage services:

- `getServiceDescription()` retrieves service information.
- `qualifyService()` discovers service and feature availability.
- `getPlanConfiguration()` retrieves plan configuration details for a service.
- `getTariffTypes()` retrieves all tariff types for a service.
- `getOverrideableDestinations()` retrieves a list of tariff destinations that can be overridden for a service.

### Related reference

Service endpoint on page 455

### `getServiceDescription()`

```
ServiceDescription getServiceDescription(int sid)
```

This method retrieves information about a service, including metadata stored in custom fields. Use this method to display service information in a web application and to access custom field metadata that describes how to display the information.

#### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID.

#### Results

This method returns a ServiceDescription XML document that contains the service name, custom field information and the subscription order form for this service.

There are no messages defined by this method.

**Note:** For more information about subscription order forms, see getAvailableOrderForms().

#### Example: A returned ServiceDescription XML document

This example shows a returned ServiceDescription XML document for an ADSL service with SID 1203.

```
<ServiceDescription xmlns="http://xml.inomial.com/smile/2.xsd">
  <SID>1203</SID>
  <name>ADSL</name>
  <ServiceProperties>
```

```
    <Object/>
  </ServiceProperties>
  <SubscriptionForm>
    <Form>
      <Name>ADSL</Name>
      <Attributes>
        <Attribute>
          <Required>false</Required>
          <Label>Stability Profile</Label>
          <Visibility>VisibleAndEditable</Visibility>
          <AttributeType>
            <AttributeType>122</AttributeType>
            <Label>Stability Profile</Label>
            <SmileObjectXMLType type="Boolean"/>
          </AttributeType>
        </Attribute>
      </Attributes>
    </Form>
  </SubscriptionForm>
</ServiceDescription>
```

### Faults

`ServiceDescriptionException`

This fault is returned when the service does not exist or the caller does not have permission to perform this action.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetServiceDescription sid
```

***sid***

**Purpose:** Specifies the service ID.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `qualifyService()`

```
ServiceQualificationResponse qualifyService(int sid,
 ServiceQualificationArguments serviceQualificationArguments)
```

This method sends data to an upstream provider for qualification. For example, this method may check the class of ADSL available at an address or check if a specific username is available at a provider.

The format and content of the data to qualify is specific to each provider. Before you use this method, contact Inomial to help you configure the service qualification module.

### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID.

**serviceQualificationArguments**

**Format:** ServiceQualificationArguments XML

**Purpose:** Defines a ServiceQualificationArguments XML document that specifies the service data to qualify. This data may vary between providers.

### Example: Request a service qualification

This example checks if a number is available at the Melbourne exchange. The fields in this document vary depending on your Smile configuration and upstream provider.

```
<ServiceQualificationArguments xmlns="http://xml.inomial.com/
smile/2.xsd">
  <arguments>
    <Object>
      <String name="fnn">+61355501256</String>
      <String name="exchange">MEL</String>
    </Object>
  </arguments>
</ServiceQualificationArguments>
```

### Results

This method returns a ServiceQualificationResponse XML document with the result of the qualification.

There are no messages defined by this method.

### Example: A returned ServiceQualificationResponse XML document

This example shows a ServiceQualificationResponse XML document that states the number requested in the previous example is available. The fields in this document vary depending on your Smile configuration and upstream provider.

```
<ServiceQualificationResponse xmlns="http://xml.inomial.com/
smile/2.xsd">
<response>
    <Object>
      <Boolean name="available">true</Boolean>
    </Object>
  </response>
</ServiceQualificationResponse>
```

### Faults

`ServiceQualificationException`

This fault is returned when the service qualification attempt fails.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.QualifyService sid filename
```

***sid***

**Purpose:** Specifies the service ID.

***filename***

**Purpose:** Defines the filename of the ServiceQualificationArguments XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `getPlanConfiguration()`

```
PlanConfiguration getPlanConfiguration(int company, string plan)
```

This method returns the plan configuration details of a service. Details returned include recurring charges, plan options, plan rules, setup charge and included value pools.

**Note:** The plan has to belong to a service.

#### Parameters

**company**

**Format:** integer

**Purpose:** Specifies the company that the plan belongs to.

**plan**

**Format:** string

**Purpose:** Specifies the plan's code.

#### Results

This method returns a PlanConfiguration XML document with the plan configuration details.

There are no messages defined by this method.

#### Example: A returned PlanConfiguration XML document

This example shows PlanConfiguration XML document containing the plan configuration details.

```
<PlanConfiguration>
  <RecurringCharges>
    <RecurringCharge id="635">
      <ChargeQuantity>1</ChargeQuantity>
      <Frequency type="Rating Periods">1</Frequency>
      <WhenToCharge>In Advance</WhenToCharge>
      <AdvancePeriods>1</AdvancePeriods>
      <ProRataAction>Pro-Rata</ProRataAction>
      <MinProRataDays>1</MinProRataDays>
      <LineItemDescription id="2629">
        <Company>1</Company>
        <ItemCode>002048</ItemCode>
        <Name>recurring charge</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$10.00">10</Rate>
        <ReportingCategory>
```

```
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
    </RecurringCharge>
  </RecurringCharges>
  <PlanOptions/>
  <ChargingRules>
    <ChargingRule id="1118">
      <ChargeType key="866">test.planChange.overrideTariffs
 Flagfall</ChargeType>
      <LineItemDescription id="2581">
        <Company>1</Company>
        <ItemCode>002000</ItemCode>
        <Name>test.planChange.overrideTariffs Flagfall</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$0.00">0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      <LineItemDescription>
      <RateSource>
        <RetailMarkup>100.00</RetailMarkup>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1119">
      <ChargeType key="867">test.planChange.overrideTariffs Timed</
ChargeType>
      <LineItemDescription id="2582">
        <Company>1</Company>
        <ItemCode>002001</ItemCode>
        <Name>test.planChange.overrideTariffs Timed</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$0.00">0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
```

```
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$0.00">0</Rate>
          <PerUnit unitQuantity="1">Second</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1122">
      <ChargeType key="868">test.planChange.overrideTariffs Pages</
ChargeType>
      <LineItemDescription id="2585">
        <Company>1</Company>
        <ItemCode>002004</ItemCode>
        <Name>test.planChange.overrideTariffs Pages</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Page</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    <ChargingRule>
    <ChargingRule id="1120">
      <ChargeType key="869">test.planChange.overrideTariffs bytes
 in</ChargeType>
      <LineItemDescription id="2583">
        <Company>1</Company>
        <ItemCode>002002</ItemCode>
```

```
        <Name>test.planChange.overrideTariffs bytes in</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Bytes</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1121">
      <ChargeType key="870">test.planChange.overrideTariffs bytes
out</ChargeType>
      <LineItemDescription id="2584">
        <Company>1</Company>
        <ItemCode>002003</ItemCode>
        <Name>test.planChange.overrideTariffs bytes out</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Bytes</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
```

```
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1123">
      <ChargeType key="871">test.planChange.overrideTariffs rate</
ChargeType>
      <LineItemDescription id="2586">
        <Company>1</Company>
        <ItemCode>002005</ItemCode>
        <Name>test.planChange.overrideTariffs rate</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Bytes/s</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1124">
      <ChargeType key="872">test.planChange.overrideTariffs bytes
 sent rate</ChargeType>
      <LineItemDescription id="2587">
        <Company>1</Company>
        <ItemCode>002006</ItemCode>
        <Name>test.planChange.overrideTariffs bytes sent rate</Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
```

```
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Bytes/s</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
    <ChargingRule id="1125">
      <ChargeType key="873">test.planChange.overrideTariffs bytes
 received rate</ChargeType>
      <LineItemDescription id="2588">
        <Company>1</Company>
        <ItemCode>002007</ItemCode>
        <Name>test.planChange.overrideTariffs bytes received rate</
Name>
        <UnitSize>1</UnitSize>
        <ChargeGst>true</ChargeGst>
        <Rate currency="AUD" formatted="$1.00">1.0</Rate>
        <ReportingCategory>
          <Company>1</Company>
          <ReportingCode>unknown</ReportingCode>
          <Name>Uncategorised Transaction</Name>
          <Id>10</Id>
        </ReportingCategory>
        <Currency>AUD</Currency>
        <Object/>
      </LineItemDescription>
      <RateSource>
        <FixedCharge>
          <Rate currency="AUD" formatted="$1.00">1.0</Rate>
          <PerUnit unitQuantity="1">Bytes/s</PerUnit>
        </FixedCharge>
      </RateSource>
      <Constraints>
        <DaysOfWeek all="true"/>
      </Constraints>
      <PrepaidRules/>
      <IncludedValuePoolRef
 key="807">test.planChange.overrideTariffs</IncludedValuePoolRef>
      <AllowOverride>false</AllowOverride>
    </ChargingRule>
  </ChargingRules>
  <IncludedValuePools>
    <IncludedValuePool id="805">
      <Name>Cap</Name>
      <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
    </IncludedValuePool>
    <IncludedValuePool id="807">
      <Name>test.planChange.overrideTariffs</Name>
```

```
      <ValueIncTax currency="AUD" formatted="$0.00">0.00</
ValueIncTax>
    </IncludedValuePool>
  </IncludedValuePools>
</PlanConfiguration>
```

### Faults

`NoSuchItemException`

This fault is returned when the plan specified does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetPlanConfiguration company
 plan
```

***company***

**Purpose:** Specifies the company that the plan belongs to.

***plan***

**Purpose:** Specifies the plan's code.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getTariffTypes()`

```
TariffTypes getTariffTypes(int sid)
```

This method returns all tariff types for a given service.

### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the ID of the service for which to return tariff types.

### Results

This method returns a TariffTypes XML document containing information about the tariff types of a service. The document can contain multiple tariff types.

There are no messages defined by this method.

### Example: A returned TariffTypes XML document

This example shows a TariffTypes XML document containing tariff types for service 772.

```
<TariffTypes>
  <TariffTypeType>
    <TariffType>1537</TariffType>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
```

```
    <Name>13XX</Name>
  </TariffTypeType>
  <TariffTypeType>
    <TariffType>1538</TariffType>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>Calls between customers</Name>
  </TariffTypeType>
  <TariffTypeType>
    <TariffType>1539</TariffType>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>Internal</Name>
  </TariffTypeType>
  <TariffTypeType>
    <TariffType>1540</TariffType>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>International</Name>
  </TariffTypeType>
  <TariffTypeType>
    <TariffType>1541</TariffType>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>Local</Name>
  </TariffTypeType>
</TariffTypes>
```

### Faults

`ServiceTariffTypeException`

This fault is returned when the service ID can not be found.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetTariffTypes sid
```

***sid***

**Purpose:** Specifies the ID of the service for which to return tariff types.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getOverrideableDestinations()`

```
Destinations getOverrideableDestinations(int sid)
```

This method returns a list of tariff destinations that can be overridden for a given service.

### Parameters

**sid**

**Format:** integer

**Purpose:** Specifies the service ID of destinations that can be overridden.

### Results

This method returns a Destinations XML document containing information about the tariff destinations that can be overridden for a service. The document can contain multiple destinations.

There are no messages defined by this method.

### Example: A returned Destinations XML document

This example shows a Destinations XML document containing the destinations that can be overridden for service 772.

```
<Destinations>
  <DestinationType>
    <Destination>67558</Destination>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>Emergency</Name>
    <DestinationClass>Domestic</DestinationClass>
  </DestinationType>
  <DestinationType>
    <Destination>67559</Destination>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>Local</Name>
    <DestinationClass>Domestic</DestinationClass>
  </DestinationType>
  <DestinationType>
    <Destination>69751</Destination>
    <Service key="772">Test service 772</Service>
    <Company key="1">Example Telecom</Company>
    <Name>my destination</Name>
    <DestinationClass>Domestic</DestinationClass>
  </DestinationType>
</Destinations>
```

### Faults

There are no faults returned by this method.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetOverrideableDestinations sid
```

***sid***

**Purpose:** Specifies the service ID of destinations that can be overridden.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## Manage number pools

Number pools are managed groups of numbers that you can reserve for certain uses, for example particular customer groups or subscriptions. A number pool defines the number range(s) the pool includes and duration settings for allocation and quarantine of a number.

A number in a pool has four allocation statuses:

- **temporary**—the number has been selected and cannot be selected elsewhere, but is not allocated until the selection has been saved.
- **pre-allocation**—the number has been allocated, but is not yet active or in use. The number cannot be allocated elsewhere.
- **allocated**—the number is in use and cannot be allocated elsewhere.
- **quarantine**—the number is no longer in use, but cannot be re-allocated until it has been released.

Methods that manage number pools are located on the pool endpoint. The pool WSDL is located at `http://smile.example.com:port/stage/smile/ws/v2/pool?wsdl`

The following methods manage number pools:

- `search()` performs a search for reservable resources across a pool.
- `getLink()` retrieves the details of a link by ID.
- `createReservation()` reserves a range of resources from the pool specified, and creates an initial link to the range.
- `createLinkToReservation()` adds a link to an existing reservation.
- `unlink()` reserves a range of resources from the pool specified, and creates an initial link to the range.

### Related reference

Pool endpoint on page 448
SmileNumberPool on page 482

### `search()`

```
PoolSearchResults search(PoolSearchCriteria criteria)
```

This method performs a search for available resources that can be reserved across a pool.

#### Parameters

**criteria**

**Format:** PoolSearchCriteria XML

**Purpose:** Defines a PoolSearchCriteria XML document that specifies the parameters of the search.

#### Example: A PoolSearchCriteria XML document

This example shows a PoolSearchCriteria XML document searching for a range of 10,000 numbers, starting from 039997000, with a maximum of 3 results to be returned.

A PoolSearchCriteria XML document can contain the following elements:

| | |
|---|---|
| **<PoolId>** | Specifies the ID of the pool to search. |
| **<SearchRangeStart>** | Specifies the resource representing the first address from which to begin the search. |
| **<SearchRangeEnd>** | Specifies the resource representing the last inclusive address to search. |
| **<DesiredRangeSize>** | (Optional) Specifies the size of the resource range to return.<br>Default=0. 0 means results are not checked for size. A positive value will return groups with the range size specified. |
| **<ResultLimit>** | (Optional) Specifies the number of pool results to return.<br>Default=30. |
| **<BackLink>** | (Optional) Specifies that results still in quarantine are included for this backLink. |

```
<PoolSearchCriteria xmlns="http://xml.inomial.com/smile/2.xsd">
  <PoolId>6070712e-0058-4114-8cbd-94a2c57d145c</PoolId>
  <SearchRangeStart>0399970000</SearchRangeStart>
  <SearchRangeEnd>0399999999</SearchRangeEnd>
  <DesiredRangeSize>10000</DesiredRangeSize>
  <ResultLimit>3</ResultLimit>
</PoolSearchCriteria>
```

### Results

This method returns a PoolSearchResults XML document listing the results of the search.

There are no messages defined by this method.

### Example: A returned PoolSearchResults XML document

This example shows a PoolSearchResults XML document. The numbers 0399990000 through 0399999999 are already reserved, so there are only two available blocks of 10,000 numbers listed.

```
<PoolSearchResults xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns:ns3="http://ws.inomial.com/smile.2">
  <PoolSearchResult>
    <PoolId>6070712e-0058-4114-8cbd-94a2c57d145c</PoolId>
    <RangeStart>0399970000</RangeStart>
    <RangeEnd>0399979999</RangeEnd>
  </PoolSearchResult>
  <PoolSearchResult>
    <PoolId>6070712e-0058-4114-8cbd-94a2c57d145c</PoolId>
    <RangeStart>0399980000</RangeStart>
    <RangeEnd>0399989999</RangeEnd>
  </PoolSearchResult>
</PoolSearchResults>
```

### Faults

`InvalidRequestException`

This fault is returned when a required parameter is not specified, or a parameter is invalid.

`NoSuchItemException`

This fault is returned when no pool with the given poolId exists.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.search filename
```

***filename***

**Purpose:** Defines the filename of the PoolSearchCriteria XML document.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `getLink()`

```
PoolReservationLink getLink(string linkId)
```

This method retrieves the details of a link by ID.

### Parameters

**linkId**

**Format:** string

**Purpose:** Specifies the ID of the link to retrieve.

### Results

This method returns a PoolReservationLink XML document containing information about the reservation link.

There are no messages defined by this method.

### Example: A returned PoolReservationLink XML document

This example shows a PoolReservationLink XML document that contains information about the reservation link.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>a9d8f38e-f9da-4f82-8396-4a0997b72435</LinkId>
  <LinkType>Allocation</LinkType>
  <Reservation>
    <ReservationId>732b30ff-824b-4a0e-9eea-691e280f2d5e</
ReservationId>
    <Start>0399990000</Start>
    <End>0399999999</End>
  </Reservation>
```

```
</PoolReservationLink>
```

### Faults

`NoSuchItemException`

This fault is returned when the link ID does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.GetLink linkId
```

***linkId***

**Purpose:** Specifies the ID of the link to retrieve.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

## `createReservation()`

```
PoolReservationLink createReservation(string poolId, string start,
 string end, LinkType linkType, BackLink backLink)
```

This method reserves a range of resources from the pool specified, and creates an initial link to the range.

### Parameters

**poolId**

**Format:** string

**Purpose:** Specifies the identifier of the pool from which to reserve.

**start**

**Format:** string

**Purpose:** Specifies the first resource of the range to reserve.

**end**

**Format:** string

**Purpose:** Specifies the last resource of the range to reserve (inclusive).

**linkType**

**Format:** LinkType XML

**Purpose:** Defines a LinkType XML document that specifies the link type for the initial link. LinkType has a value of Temporary, PreAllocation, Allocation or Quarantine.

**backLink**

**Format:** BackLink XML

**Purpose:** (Optional) Defines a BackLink XML document that specifies a backlink identifying the owner of the link.

### Results

This method returns a PoolReservationLink XML document containing the a link to the new reservation.

There are no messages defined by this method.

### Example: A returned PoolReservationLink XML document

This example shows PoolReservationLink XML document containing details of a new temporary reservation of 10,000 numbers starting at 039999.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>c71e5fb9-f467-41a7-aa6e-d82d4ce7f7c6</LinkId>
  <LinkType>Temporary</LinkType>
  <Reservation>
    <ReservationId>732b30ff-824b-4a0e-9eea-691e280f2d5e</
ReservationId>
    <Start>0399990000</Start>
    <End>0399999999</End>
  </Reservation>
  <Expiry>2015-02-08T14:47:52.163+11:00</Expiry>
</PoolReservationLink>
```

### Faults

`InvalidRequestException`

This fault is returned when a required field is not specified, linkType is invalid, or start and/or end are invalid.

`NoSuchItemException`

This fault is returned when one of the referenced objects does not exist

`ReservationUnavailableException`

This fault is returned when any of the desired resources are already reserved.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateReservation poolId start
 end filename [backLinkType] [backLinkKey]
```

***poolId***

**Purpose:** Specifies the identifier of the pool from which to reserve.

***start***

**Purpose:** Specifies the first resource of the range to reserve.

***end***

**Purpose:** Specifies the last resource of the range to reserve (inclusive).

***filename***

**Purpose:** Defines the filename of the LinkType XML document.

***backLinkType***

**Purpose:** (Optional) Specifies a unique string representing the type of object that is holding the reference.

***backLinkKey***

**Purpose:** (Optional) Specifies a string uniquely identifying object that is holding the reference, within the scope of the backLinkType.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `createLinkToReservation()`

```
PoolReservationLink createLinkToReservation(string reservationId,
 LinkType linkType, BackLink backLink)
```

This method adds a link to an existing reservation.

#### Parameters

**reservationId**

**Format:** string

**Purpose:** Specifies the ID of the reservation to link to.

**linkType**

**Format:** LinkType XML

**Purpose:** Defines a LinkType XML document that specifies the link type for the initial link. LinkType has a value of Temporary, PreAllocation, Allocation or Quarantine.

**backLink**

**Format:** BackLink XML

**Purpose:** (Optional) Defines a BackLink XML document that specifies a backlink identifying the owner of the link.

#### Results

This method returns a PoolReservationLink XML document containing the a link to the new reservation.

There are no messages defined by this method.

#### Example: A returned PoolReservationLink XML document

This example shows a PoolReservationLink XML document returned after adding an Allocation link type to a reservation.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>a9d8f38e-f9da-4f82-8396-4a0997b72435</LinkId>
  <LinkType>Allocation</LinkType>
  <Reservation>
```

```
    <ReservationId>732b30ff-824b-4a0e-9eea-691e280f2d5e</
ReservationId>
    <Start>0399990000</Start>
    <End>0399999999</End>
  </Reservation>
</PoolReservationLink>
```

### Faults

`InvalidRequestException`

This fault is returned when a required field is not specified, linkType is invalid, or the reservation is no longer active.

`NoSuchItemException`

This fault is returned when one of the referenced objects does not exist.

### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.CreateLinkToReservation
 reservationId filename [backLinkType] [backLinkKey]
```

***reservationId***

**Purpose:** Specifies the ID of the reservation to link to.

***filename***

**Purpose:** Defines the filename of the LinkType XML document.

***backLinkType***

**Purpose:** (Optional) Specifies a unique string representing the type of object that is holding the reference.

***backLinkKey***

**Purpose:** (Optional) Specifies a string uniquely identifying object that is holding the reference, within the scope of the backLinkType.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

### `unlink()`

```
PoolReservationLink unlink(string linkId)
```

This method removes a resource link.

If the link ID is a concrete link (Pre-Allocation or Allocation) then a Quarantine link will be created and returned if Smile is configured to do so.

#### Parameters

**linkId**

**Format:** string

**Purpose:** Specifies the ID of the link to remove.

#### Results

This method returns a quarantine link if one is created, otherwise no results are returned.

Smile emits a `LinkDeallocation` message when Smile removes the resource link.

#### Example: A returned PoolReservationLink XML document

This example shows a PoolReservationLink XML document that contains a quarantine link type.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>e55fe410-5d6f-43f3-bbf1-51bafb2f07fe</LinkId>
  <LinkType>Quarantine</LinkType>
  <Reservation>
    <ReservationId>732b30ff-824b-4a0e-9eea-691e280f2d5e</
ReservationId>
    <Start>0399990000</Start>
    <End>0399999999</End>
  </Reservation>
  <Expiry>2015-04-09T13:34:13.073+10:00</Expiry>
</PoolReservationLink>
```

#### Faults

`NoSuchItemException`

This fault is returned when the given linkId does not exist.

#### Java client syntax

```
java -cp build/smilewsv2-client.jar [options]
 com.inomial.smile.client.v2.examples.Unlink linkId
```

***linkId***

**Purpose:** Specifies the ID of the link to remove.

**Note:** For more information about the Java client and the options it accepts, see Java client syntax.

# Appendix A

# SOAP methods

## Account endpoint

This topic provides a quick reference for the method signatures of account API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/account`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/account?wsdl`

**Table 4: Account API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addDeferredCharge() | `addDeferredCharge( string usn, ChargeRequest charge)` | Adds deferred item charges to an account. |
| allocate() | `void allocate( string usn, AllocationRequests allocationRequests)` | Allocates transactions within an account. |
| compileChargeList() | `NewInvoiceRequest compileChargeList( string usn, ChargeListCompilationRequest chargeListCompilationRequest)` | Compiles a quote of charges for an account. |
| createBond() | `ItemisedTransactionDetail createBond( BondRequest bondRequest)` | Creates a bond on an account. |
| createCostCentre() | `CostCentree createCostCentre( NewCostCentre newCostCentre)` | Creates a new cost centre for an account. |
| createPurchaseOrder() | `PurchaseOrder createPurchaseOrder( NewPurchaseOrderRequest newPurchaseOrderRequest)` | Returns a purchase order document that has been persisted. |
| deletecCostCentre() | `void deleteCostCentre( int costCentreId)` | Deletes an unused cost centre form an account. |
| echo() | `string echo(string value)` | Returns any value passed to the method. |

| Method | Method signature | Purpose |
|---|---|---|
| findAccount() | `Account findAccount( SearchCondition searchCondition)` | Searches by custom field value for an account and returns account information. |
| findAccounts() | `AccountSearchResultList findAccounts( SearchCondition searchCondition, int limit, int offset)` | Searches by custom field for non-closed accounts which are not subscriptions and returns information about any matches. |
| findCostCentre() | `CostCentres findCostCentre( string usn, string costCentreName)` | Searches cost centres of an account by cost centre name. |
| get() | `Account get(string usn)` | Searches by USN for an account and returns account and subscription information. |
| getAccountInfo() | `Account getAccountInfo(string usn)` | Searches by USN for an account and returns account information. |
| getAccountSummary() | `AccountSummary getAccountSummary( string usn)` | Searches by USN for an account and returns the account balance, treatment level, aged balances and the number of active subscriptions on the account. |
| getBondTypes() | `BondTypes getBondTypes()` | Returns the bond types that are available in Smile. |
| getCostCentre() | `CostCentre getCostCentre( int costCentreId)` | Retrieves cost centre information for a given ID. |
| getCostCentres() | `CostCentres getCostCentres( string usn)` | Returns a list of cost centres that are associated with an account. |

| Method | Method signature | Purpose |
|---|---|---|
| getDeferredCharges() | `DeferredChargeList getDeferredCharges( string usn)` | Returns a list of deferred charges for a given USN. |
| getFutureBilling() | `FutureInvoiceList getFutureBilling( string usn, FutureInvoiceRequest request)` | Returns a future invoice list for an account, including subscriptions, for given invoicing end date(s). |
| getHeldBonds() | `Bonds getHeldBonds( string usn)` | Returns a list of bonds on an account, including released bonds. |
| getPurchaseOrder() | `PurchaseOrder getPurchaseOrder( string purchaseOrderId)` | Returns the account purchase order for the given ID. |
| getRatings() | `ActivityBatchURL getRatings( string transactionNumber)` | Returns a URL from which you can download the rated activity statements for a given invoice. |
| getTransaction() | `TransactionDetail getTransaction( string transactionNumber)` | Searches by transaction number for a transaction and returns transaction information. |
| getTransactionDocument() | `DocumentReferenceType getTransactionDocument( string transactionNumber)` | Searches by transaction number for a transaction and returns transaction information in a rendered document format. |
| getTransactionList() | `TransactionList getTransactionList( string usn, dateTime startTimestamp, dateTime endTimestamp)` | Returns a list of transactions on an account. |

| Method | Method signature | Purpose |
|---|---|---|
| listPurchaseOrders() | `PurchaseOrderList listPurchaseOrders( string account)` | Returns a list of purchase orders for an account. |
| migrateOpenTransaction() | `ItemisedTransactionDetail migrateOpenTransaction( string openTransactionNumber, string targetAccountUsn)` | Moves an open transaction to another account. |
| raiseAdjustment() | `TransactionDetail raiseAdjustment( string usn, int adjustmentType, dateTime effective, decimal amount, decimal gst, string comment, AllocationRequest allocationRequest)` | Raises a credit or debit adjustment against an account. |
| raiseInvoice() | `ItemisedTransactionDetail raiseInvoice( string usn, NewInvoiceRequest newInvoiceRequest)` | Raises an invoice to an account. |
| raiseReimbursement() | `TransactionDetail raiseReimbursement( string usn, ReimbursementRequest reimbursementRequest)` | Raises a reimbursement to an account. |
| releaseBond() | `TransactionDetail releaseBonds( string usn, string bondNumber)` | Releases a bond from an account. |

| Method | Method signature | Purpose |
|---|---|---|
| reverseTransaction() | `TransactionDetail reverseTransaction( string usn, ReverseTransactionRequest reverseTransactionRequest)` | Reverses a transaction. |
| update() | `Account update( string usn, Account account)` | Updates account information. |
| updateCostCentre() | `void updateCostCentre( CostCentre costCentre)` | Updates cost centre information on an account. |
| updatePurchaseOrder() | `void updatePurchaseOrder( PurchaseOrder purchaseOrder)` | Updates the details of a purchase order. |

Related concepts

Account API overview on page 75

## Accounts receivable endpoint

This topic provides a quick reference for the method signatures of accounts receivable API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/accountsreceivable`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/accountsreceivable?wsdl`

**Table 5: Accounts receivable API methods**

| Method | Method signature | Purpose |
|---|---|---|
| getItemDescription() | `LineItemDescription getItemDescription( int company, string itemCode)` | Returns an item description for the given company and item code. |
| getItemDescriptionByID() | `LineItemDescription getItemDescriptionById( string itemDescriptionId)` | Returns details of an item description. |

Related concepts

Manage accounts receivable on page 384

## Authentication endpoint

This topic provides a quick reference for the method signatures of authentication API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/authentication`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/authentication?wsdl`

**Table 6: Authentication API methods**

| Method | Method signature | Purpose |
|---|---|---|
| authenticate() | `AuthenticationResponse authenticate( AuthenticationRequest authenticationRequest)` | Authenticates subscription credentials. |

Related concepts

Authenticate credentials on page 382

## CAT endpoint

This topic provides a quick reference for the method signatures of CAT API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/cat`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/cat?wsdl`

**Table 7: Authentication API methods**

| Method | Method signature | Purpose |
|---|---|---|
| triggerCatForgot PasswordEmail() | `void triggerCatForgotPasswordEmail( string catUsername)` | Returns and emails CAT user password details. |

Related concepts

Manage CAT on page 387

## Contract endpoint

This topic provides a quick reference for the method signatures of contract API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/contract`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/contract?wsdl`

**Table 8: Contract API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addSubscriptionContract() | `SubscriptionContract addSubscriptionContract( string usn, int contractId, dateTime startDate, dateTime endDate)` | Adds a contract to a subscription. |
| commitContractTermination() | `SubscriptionContractTerminationResponse commitContractTermination( string usn, SubscriptionContractTerminationOffer subscriptionContractTerminationOffer)` | Terminate a contract on a subscription. |
| getActiveSubscriptionContract() | `SubscriptionContract getActiveSubscriptionContract( string usn, dateTime when)` | Returns details about a contract on a subscription. |
| getAvailableContractsForPlan() | `PlanContractDescriptionList getAvailableContractsForPlan( int company, int plan, dateTime when)` | Returns contracts available to subscriptions on a particular plan. |
| getContractsForService() | `ContractDescriptionList getContractsForService( int sid)` | Returns contracts on a particular service. |
| getCurrentSubscriptionContract() | `SubscriptionContract getCurrentSubscriptionContract( string usn)` | Returns the current contract for a subscription. |

| Method | Method signature | Purpose |
|---|---|---|
| getSubscriptionContractById() | `SubscriptionContract getSubscriptionContractById( string subscriptionContractId)` | Returns a subscription contract by ID. |
| getSubscriptionContracts() | `SubscriptionContractList getSubscriptionContracts( string usn)` | Returns a lists of all contracts for a subscription. |
| proposeContractTermination() | `SubscriptionContractTerminationOffer proposeContractTermination( string usn, int subscriptionContractId, dateTime terminationDate)` | Propose a contract termination on a subscription. |
| updateSubscriptionContract() | `void updateSubscriptionContract( string usn, SubscriptionContract subscriptionContract)` | Updates a subscription contract with new values. |

Related concepts

Contract API overview on page 329

## CRM endpoint

This topic provides a quick reference for the method signatures of CRM API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/crm`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/crm?wsdl`

**Table 9: CRM API methods**

| Method | Method signature | Purpose |
|---|---|---|
| echo() | `string echo(string value)` | Returns any value passed to the method. |
| addTicketAttachment() | `void addTicketAttachment( string ticketNumber, Attachment attachment)` | Adds an attachment to a ticket. |
| getAppointmentAvailabilities() | `AppointmentAvailability getAppointmentAvailabilities( AppointmentAvailabilitiesRequest availabilitiesRequest)` | Retrieves appointment availabilities. |
| getTemplateTicketDescription() | `TemplateTicketDescription getTemplateTicketDescription( string templateTicketNumber)` | Returns the details of a template ticket. |
| getTicket() | `Ticket getTicket( string ticketNumber)` | Retrieves ticket information. |
| getTickets() | `TicketList getTickets( string usn)` | Retrieves all tickets for an account. |
| raiseTicket() | `Ticket raiseTicket( string templateTicketNumber, string usn, NewTicketRequest newTicketRequest)` | Creates a ticket and starts a workflow. |
| runTicketAction() | `Ticket runTicketAction( string ticketNumber, int ticketAction)` | Retrieves information about a ticket after running a ticket action. |

| Method | Method signature | Purpose |
|---|---|---|
| updateTicket() | `void updateTicket( string ticketNumber, TicketUpdate ticketUpdate)` | Updates a tickets' subject, body text, custom fields or owning account or subscription details. |

Related concepts

CRM API overview on page 360

## Custom fields endpoint

This topic provides a quick reference for the method signatures of custom fields API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/config/customfields`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/config/customfields?wsdl`

**Table 10: Custom fields API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addCollectionEntry() | `CustomCollectionEntry addCollectionEntry( NewCustomCollectionEntryRequest newEntryRequest)` | Creates a new collection entry for a custom collection. |
| deleteCollectionEntry() | `void deleteCollectionEntry( int key)` | Deletes a custom collection entry. |
| findCollectionEntry() | `CustomCollectionEntrySearchResults findCollectionEntries( string collectionCode, SearchCondition criteria, int limit, int offset)` | Searches a custom collection custom fields entries that match a given value. |
| getCollectionEntry() | `CustomCollectionEntry getCollectionEntry( int key)` | Retrieves a custom collection entry directly by key. |
| getCollectionEntryByEntryCode() | `CustomCollectionEntry getCollectionEntryByEntryCode( string collectionCode, string entryCode)` | Retrieves details of a custom collection entry by entry code. |
| updateCollectionEntry() | `void updateCollectionEntry( CustomCollectionEntry entry)` | Updates an existing custom collection entry. |

Related concepts

Custom fields API overview on page 349

## Invoice grouping endpoint

This topic provides a quick reference for the method signatures of invoice grouping API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/account/invoicegrouping`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/account/invoicegrouping?wsdl`

**Table 11: Invoice grouping API methods**

| Method | Method signature | Purpose |
|---|---|---|
| createInvoiceGrouping() | `void createInvoiceGrouping( NewInvoiceGrouping newInvoiceGroupingRequest)` | Creates a new invoice grouping. |
| deleteInvoiceGrouping() | `void deleteInvoiceGrouping( string invoiceGroupingId)` | Deletes an invoice grouping. |
| getAvailableInvoiceGrouping() | `InvoiceGroupings getAvailableInvoiceGroupings( string usn)` | Retrieves the list of invoice groupings configured to an account. |
| getInvoiceGrouping() | `InvoiceGroupings getInvoiceGrouping( string invoiceGroupingId)` | Retrieves the invoice grouping for an ID. |
| updateInvoiceGrouping() | `void updateInvoiceGrouping( string invoiceGroupingId, NewInvoiceGrouping updateInvoiceGroupingRequest)` | Updates an invoice grouping. |

Related concepts

Manage invoice grouping on page 131

## Managed plans endpoint

This topic provides a quick reference for the method signatures of managed plans API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/subscription/plan`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/subscription/plan?wsdl`

**Table 12: Managed plans API methods**

| Method | Method signature | Purpose |
|---|---|---|
| changeManagedPlanGroup() | `void changeManagedPlanGroup( string usn, string managedPlanGroupId, string managedPlanChoiceId, ManagedPlanChangeTiming managedPlanChangeTiming, PlanOptions planOptions)` | Changes the managed plan group that the subscription belongs to. |
| changePlan() | `ManagedPlanChangeResponse changePlan( string usn, string managedPlanChoiceId, PaymentRequest paymentRequest, PlanOptions planOptions)` | Changes the plan of the provided subscription. |
| getCurrentManagedPlanGroup() | `ManagedPlanGroup getCurrentManagedPlanGroup( string usn)` | Returns the details of the managed plan group that the subscription provided is in. |
| getManagedPlanChoices() | `ManagedPlanQuoteList getManagedPlanChoices( string usn)` | Returns a list of possible managed plan changes. |
| getManagedPlanGroup() | `ManagedPlanGroup getManagedPlanGroup( string managedPlanGroupId)` | Returns the details of the managed plan group specified. |
| getManagedPlanGroups() | `ManagedPlanGroupList getManagedPlanGroups( int sid)` | Returns a list of managed plan groups, and their choices, for the given service. |

| Method | Method signature | Purpose |
|---|---|---|
| getPlanSchedule() | `ManagedPlanSchedule getPlanSchedule( string usn)` | Returns the current and future managed plan changes. |

Related concepts

Managed plans API overview on page 244

## Order endpoint

This topic provides a quick reference for the method signatures of order API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/order`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/order?wsdl`

**Table 13: Order API methods**

| Method | Method signature | Purpose |
|---|---|---|
| approveOrder() | `void approveOrder( string orderNumber)` | Approves an order and progresses it to the next stage in the order process. |
| addOrderToGroup() | `OrderGroup addOrderToGroup( string orderGroupId, Order order)` | Adds the order specified to the order group. |
| cancelOrder() | `void cancelOrder( string orderNumber)` | Cancels the given order and removes it from the group. |
| cancelOrderGroup() | `void cancelOrderGroup( string orderGroupNumber)` | Cancels all orders in an order group. |
| createOrderGroup() | `OrderGroup createOrderGroup( Order initialOrder)` | Creates a new order group with the initial order provided. |
| echo() | `string echo(string value)` | Returns any value passed to the method. |
| getAvailableOrderForms() | `OrderFormList getAvailableOrderForms()` | Returns order form information for each package an application is authorised to order. |
| getOrder() | `OrderDetails getOrder( string orderNumber)` | Searches by order number for an order and returns order information. |
| getOrderForm() | `OrderForm getOrderForm( int packageId)` | Returns order form information for a specific package. |

| Method | Method signature | Purpose |
|---|---|---|
| getOrderGroup() | `OrderGroup getOrderGroup( string orderGroupId)` | Returns the order group. |
| getOrderGroupQuote() | `ItemisedTransactionDetail getOrderGroupQuote( string orderGroupId)` | Retuers the quote of the given order group. |
| orderAccount() | `OrderDetails orderAccount( Order order)` | Orders a package that contains an account and may contain one or more subscriptions. |
| orderSubscription() | `OrderDetails orderSubscription( string usn, Order order)` | Orders a package that contains only subscriptions. |
| placeOrderGroup() | `OrderPlacementResponse placeOrderGroup( string orderGroupId, PaymentRequest paymentDetails)` | Submits all the orders in the order group, and may pay the initial invoice. |
| reserveUsns() | `UsnList reserveUsns( int sid, int count)` | Reserves subscription USNs that can be used for ordering later. |
| updateOrder() | `void updateOrder( string orderNumber, Order order)` | Updates order information. |

## Related concepts

Order API overview on page 37

## Payment endpoint

This topic provides a quick reference for the method signatures of payment API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/payment`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/payment?wsdl`

**Table 14: Payment API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addStoredPaymentDetails() | `StoredPaymentDetails addStoredPaymentDetails( string usn, PaymentDetails paymentDetails)` | Adds stored payment details to an account. |
| cancelScheduledPayment() | `Payment cancelScheduledPayment( string paymentNumber)` | Cancels a payment. |
| computeSurcharge() | `SurchargeCalculation computeSurcharge( string usn, string paymentTypeCode, CurrencyValue amount, date timestamp)` | Calculates the surcharge payable on a payment. |
| deleteStoredPaymentDetails() | `void deleteStoredPaymentDetails( string usn, string storedPaymentDetailsId)` | Deletes the stored payment details on an account. |
| getAutoPaymentOption() | `getAutoPaymentOptionResponse getAutoPaymentOption( string usn)` | Returns the configured automatic payment option for an account. |
| getPayment() | `Payment getPayment( string paymentNumber)` | Returns information about a payment for a specified payment number. |
| getPaymentTypes() | `PaymentTypes getPaymentTypes( string currency)` | Returns information about the payment methods appropriate for the current user. |

| Method | Method signature | Purpose |
|---|---|---|
| getPendingPayments() | `PaymentListType getPendingPayments( string accountUsn)` | Returns a list of all pending payments for a particular account. |
| getStoredPaymentDetails() | `StoredPaymentDetails getStoredPaymentDetails( string usn, string storedPaymentDetailsId)` | Returns the stored payment details for an account and ID. |
| getStoredPaymentdetailsList() | `StoredPaymentDetailsList getStoredPaymentDetailsList( string usn)` | Returns all stored payment details for an account. |
| processPayment() | `PaymentResponse processPayment( string usn, PaymentRequest paymentRequest, AllocationRequest allocationRequest)` | Processes a payment and queues the debits to be allocated to the payment, pending the payment's success. |
| recordFailedPayment() | `Payment recordFailedPayment( string usn, PaymentRequest paymentRequest, string message)` | Records failed payment to a Smile account. |
| reschedulePaymentWithNewDetails() | `Payment reschedulePaymentWithNewDetails( string paymentNumber, PaymentDetailsRequest request, date timestamp)` | Reschedules a payment to a specified time with specified new payment details. |
| updateStoredPaymentDetails() | `StoredPaymentDetails updateStoredPaymentDetails( string usn, string storedPaymentDetailsId, PaymentDetails paymentDetails)` | Updates the stored payment details on an account. |

| Method | Method signature | Purpose |
|---|---|---|
| validatePayment() | `PaymentDetailsValidationResponse validatePayment( PaymentDetailsValidationRequest paymentDetailsValidationRequest)` | Validates credit card details. |

Related concepts

Payments API overview on page 281

## Plan change endpoint

This topic provides a quick reference for the method signatures of plan change API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/subscription/planchange`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/subscription/planchange?wsdl`

**Table 15: Plan change API methods**

| Method | Method signature | Purpose |
|---|---|---|
| getServicePlans() | `PlanConfigurations getServicePlans( int sid, string usn, date timestamp)` | Returns a list of published plans available against a service or plans available to subscriptions of accounts of a given service. |
| getSubscriptionPlans() | `PlanConfigurations getSubscriptionPlans( string usn, date timestamp)` | Returns a list of published plans available to a subscription. |
| commitPlanChange() | `PlanChangeResponse commitPlanChange( string usn, PlanChangeOffer planChangeOffer)` | Changes a plan based on a previously acquired `PlanChangeOffer()`. |
| proposePlanChange() | `PlanChangeOffer proposePlanChange( string usn, PlanChangeProposal planChangeProposal)` | Returns information about a proposed plan change. |

Related concepts

Manage real-time rating plan changes on page 196

## Pool endpoint

This topic provides a quick reference for the method signatures of pool API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/pool`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/pool?wsdl`

**Table 16: Pool API methods**

| Method | Method signature | Purpose |
|---|---|---|
| createLinkToReservation() | `PoolReservationLink createLinkToReservation( string reservationId, LinkType linkType, BackLink backLink)` | Adds a link to an existing reservation. |
| createReservation() | `PoolReservationLink createReservation( string poolId, string start, string end, LinkType linkType, BackLink backLink)` | Reserves a range of resources from the pool specified, and creates an initial link to the range. |
| getLink() | `PoolReservationLink getLink( string linkId)` | Retrieves the details of a link by ID. |
| search() | `PoolSearchResults search( PoolSearchCriteria criteria)` | Performs a search for available resources that can be reserved across a pool. |
| unlink() | `PoolReservationLink unlink( string linkId)` | Removes the given link. |

## Related concepts

Manage number pools on page 411

## Prepaid endpoint

This topic provides a quick reference for the method signatures of prepaid API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/prepaid`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/prepaid?wsdl`

**Table 17: Prepaid API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addPrepaid() | `void addPrepaid( string usn, string prepaidCode, AddPrepaidRequestOverride addPrepaidRequestOverride)` | Adds a prepaid block to a subscription, but does not charge for the prepaid or raise an invoice. |
| addSelfServePrepaid() | `void addSelfServePrepaid( string usn, string prepaidCode, PaymentRequest paymentRequest)` | Adds a self-serve prepaid block to a subscription and charges for the block. |
| getPrepaidPaymentTypes() | `PaymentTypes getPrepaidPaymentTypes( string usn, string prepaidCode)` | Returns a list of payment methods that can be used to buy self-serve prepaid blocks. |
| getPrepaidState() | `SubscriptionPrepaidState getPrepaidState( string usn)` | Returns a list of prepaid blocks and their current state by USN. |
| getQuote() | `ItemisedTransactionDetail getQuote( string usn, string prepaidCode, string paymentTypeCode)` | Returns the cost to buy a self-serve prepaid block with a specific payment method. |
| getSelfServePrepaidList() | `SelfServePrepaidList getSelfServePrepaidList( string usn)` | Returns a list of self-serve prepaid blocks available to purchase. |

| Method | Method signature | Purpose |
|---|---|---|
| redeemVoucher() | `void redeemVoucher( string usn, string voucherNumber, string voucherCode)` | Consumes a voucher and adds prepaid to a subscription. |
| updatePrepaid() | `void updatePrepaid( string usn, PrepaidUpdate prepaidUpdate)` | Updates a purchased prepaid block on a subscription. |
| validateVoucher() | `VoucherValidationResponse validateVoucher( int company, string voucherNumber, string voucherCode)` | Verifies that a voucher is valid. |

## Related concepts

Prepaid API overview on page 312

## Provisioning endpoint

This topic provides a quick reference for the method signatures of provisioning API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/provisioning`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/provisioning?wsdl`

**Table 18: Provisioning API methods**

| Method | Method signature | Purpose |
|---|---|---|
| getIncompleteProvisionRequests() | `ProvisionRequests getIncompleteProvisionRequests( string usn)` | This method retrieves a list of currently incomplete provisioning requests that have been lodged for a subscription. |
| provisioningUpdate() | `ProvisioningResponse provisioningUpdate( int provisionRequestId, ProvisionRequestStatus status, string logMessage)` | This method updates the provision status of a provision request with a log message. |

Related concepts

Manage provisioning on page 388

## Recurring charge endpoint

This topic provides a quick reference for the method signatures of recurring charge API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/recurringcharge`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/recurringcharge?wsdl`

**Table 19: Prepaid API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addSimpleRecurringCharges() | `SimpleRecurringCharges addSimpleRecurringCharges( string usn, NewSimpleRecurringCharges recurringCharges)` | Adds recurring charges to a specified account or subscription. |
| getSimpleRecurringCharge() | `SimpleRecurringCharge getSimpleRecurringCharge( string userAutoChargeId)` | Retrieves a recurring charge with the specified recurring charge ID. |
| getSimpleRecurringCharges() | `SimpleRecurringCharges getSimpleRecurringCharges( string usn)` | Retrieves a list of recurring charges for a specified account or subscription. |
| updateSimpleRecurringCharges() | `void updateSimpleRecurringCharges( string usn, NewSimpleRecurringCharges recurringCharges)` | Updates recurring charges of a specified account or subscription. |

Related concepts

Manage recurring charges on page 392

## Service endpoint

This topic provides a quick reference for the method signatures of service API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/service`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/service?wsdl`

**Table 20: Service API methods**

| Method | Method signature | Purpose |
|---|---|---|
| echo() | `string echo(string value)` | Returns any value passed to the method. |
| getOverrideableDestinations() | `Destinations getOverrideableDestinations( int sid)` | Returns a list of tariff destinations that can be overridden for a given service. |
| getPlanConfiguration() | `PlanConfiguration getPlanConfiguration( int company, string plan)` | Returns the plan configuration details of a service. |
| getServiceDescription() | `ServiceDescription getServiceDescription( int sid)` | Retrieves service information. |
| getTariffTypes() | `TariffTypes getTariffTypes( int sid)` | Returns all tariff types for a given service. |
| qualifyService() | `ServiceQualificationResponse qualifyService( int sid, ServiceQualificationArguments serviceQualificationArguments)` | Sends data to an upstream provider for qualification. |

### Related concepts

Manage services on page 399

## Spend management endpoint

This topic provides a quick reference for the method signatures of spend management API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/spendmanagement`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/spendmanagement?wsdl`

**Table 21: Spend management API methods**

| Method | Method signature | Purpose |
|---|---|---|
| getSubscriptionSpendLimitState() | `SubscriptionSpendLimit getSubscriptionSpendLimitState( string usn, string spendLimitId)` | Returns the current state of a specified spend limit for a subscription. |
| getSubscriptionSpendLimitStates() | `SubscriptionSpendLimitState getSubscriptionSpendLimitStates( string usn)` | Returns the current state of all spend limits for a subscription. |
| getSubscriptionUsageAlertGroup() | `SubscriptionUsageAlertGroup getSubscriptionUsageAlertGroup( string usn, string usageAlertGroupId)` | Returns the current state of a specified usage alert group for a subscription. |
| getSubscriptionUsageAlertGroups() | `SubscriptionUsageAlertGroups getSubscriptionUsageAlertGroups( string usn)` | Returns the current state of all usage alert groups for a subscription. |
| getSubscriptionValuePoolState() | `SubscriptionValuePool getSubscriptionValuePoolState( string usn, string valuePoolId)` | Returns the current state of a specified value pool for a subscription. |
| getSubscriptionValuePoolStates() | `SubscriptionValuePoolState getSubscriptionValuePoolStates( string usn)` | Returns the current state of all value pools for a subscription. |

| Method | Method signature | Purpose |
|---|---|---|
| setSpendLimitAlertThresholdOverrides() | `void setSpendLimit AlertThresholdOverrides( string usn, SubscriptionSpendLimitAlertThresholdOverrideRequest subscriptionSpendLimitAlertThresholdOverrideRequest)` | Changes the alert thresholds for a subscription spend limit. |
| setSpendLimitOverride() | `void setSpendLimitOverride( string usn, SubscriptionSpendLimitOverrideRequest subscriptionSpendLimitOverrideRequest)` | Permanently changes the value of a subscription spend limit. |
| setTemporarySpendLimitOverride() | `void setTemporarySpend LimitOverride( string usn, TemporarySubscriptionSpendLimitOverrideRequest temporarySubscriptionSpendLimitOverrideRequest)` | Temporarily changes the value of a subscription spend limit. |

Related concepts

Spend management API overview on page 262

## Subscription endpoint

This topic provides a quick reference for the method signatures of subscription API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/subscription`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/subscription?wsdl`

**Table 22: Subscription API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addIpRange() | `SubscriptionIpRange addIpRange( string usn, IpRange ipRange)` | Allocates an IP range reservation to a subscription. |
| addIpRangeReservation() | `SubscriptionIpRange addIpRangeReservation( string usn, string reservationId)` | Adds an IP range reservation to a subscription. |
| applyFeature() | `subscriptionFeatureId applyFeature( string usn, int featureId, date startDate, date endDate)` | Applies a feature to a subscription for the specified length of time. |
| applyFeatureToConnections() | `void applyFeatureToConnections( string usn, int featureId)` | Applies a feature to any active connections for a USN. |
| cancel() | `void cancel( string usn, date fromTimestamp)` | Cancels a subscription. |
| changeUsername() | `void changeUsername( string usn, string username)` | Changes the username of a subscription. |
| changePlan() | `void changePlan( string usn, PlanScheduleItem planScheduleItem)` | Changes the plan schedule for a subscription, that is, changes the plan that a subscription uses. |

| Method | Method signature | Purpose |
|---|---|---|
| findSubscription() | `Subscription findSubscription( SearchCondition searchCondition)` | Searches by custom field value for a subscription and returns subscription information. |
| findSubscriptions() | `SubscriptionSearchResultList findSubscriptions( SearchCondition searchCondition, int limit, int offset)` | Searches by custom field for non-closed subscriptions and returns subscription information. |
| findUsername() | `string findUsername( int sid, string username)` | Returns the USN for a given username and service. |
| get() | `Subscription get(string usn)` | Searches by USN for a subscription and returns subscription information. |
| getActiveFeatures() | `SubscriptionFeatureList getActiveFeatures( string usn)` | Returns all active features on a subscription. |
| getActivityStatementBatch() | `ActivityBatchURL getActivityStatementBatch( string usn, long batchId)` | Returns a URL from which you can download unrated activity statements by batch number. |
| getActivityStatementRange() | `ActivityBatchURL getActivityStatementRange( string usn, dateTime startTimestamp, dateTime endTimestamp)` | Returns a URL from which you can download unrated activity statements by date range. |

| Method | Method signature | Purpose |
|---|---|---|
| getAuthenticationCredentials() | `Credentials getAuthenticationCredentials( string usn)` | Retrieves a user password. |
| getConnectionHistory() | `ConnectionListType getConnectionHistory( string usn, ConnectionHistoryQuery connectionHistoryQuery, int limit, int offset)` | Returns the connection history of a subscription for a given date range. |
| getIpRanges() | `SubscriptionIpRanges getIpRanges( string usn)` | Returns all IP range reservations for a subscription. |
| getPlanChargeDescription() | `PlanChargeDescriptionType getPlanChargeDescription( string usn, int planScheduleItem)` | Returns information about the charge configuration for a subscription, according to the given plan schedule item. |
| getPlanScheduleItemsAt() | `PlanScheduleItems getPlanScheduleItemsAt( string usn, timestamp instant)` | Returns information about the plan that a subscription uses at a specified time. |
| getPlanSchedules() | `PlanSchedules getPlanSchedules( string usn)` | Returns information about all plans associated with a subscription throughout its lifecycle. |
| getPoolReservationLinks() | `PoolReservationLinks getPoolReservationLinks( string usn)` | Returns all active pool reservation links of a subscription. |

| Method | Method signature | Purpose |
|---|---|---|
| getRatingPeriods() | `RatingPeriodListType getRatingPeriods(string usn)` | Returns information about a subscription's rating periods. |
| getRatingsForPeriod() | `ActivityBatchURL getRatingsForPeriod( int ratingPeriodId)` | Returns a URL from which you can download rated activity statements by rating period. |
| migrateSubscription() | `void migrateSubscription( string usn, string toAccountUsn)` | Migrates a subscription from one account to another nominated account. |
| removeIpRange() | `void removeIpRange( string usn, string ipRangeId)` | Removes an IP range reservation that was previously made with `addIpRange()`. |
| removePoolReservationLink() | `PoolReservationLink removePoolReservationLink( string usn, string reservationLinkId)` | Removes a reservation link applied to the subscription. |
| resume() | `void resume( string usn, date fromTimestamp, string comment)` | Resumes a suspended subscription. |
| setClearPassword() | `void setClearPassword( string usn, Credential credential)` | Changes the password for an account or subscription. |
| setCostCentre() | `void setCostCentre( string usn, int costCentreId)` | Sets a cost centre for a subscription. |

| Method | Method signature | Purpose |
|---|---|---|
| stopApplyingFeature() | `void stopApplyingFeature(`<br>` string subscriptionFeatureId,`<br>` date endDate)` | Changes the date on which Smile stops applying a feature. |
| suspend() | `void suspend(`<br>` string usn,`<br>` date fromTimestamp,`<br>` string comment)` | Suspends a subscription. |
| update() | `Subscription update(`<br>` string usn,`<br>` Subscription subscription)` | Updates subscription information. |

Related concepts

Subscription API overview on page 156

## Subservice endpoint

This topic provides a quick reference for the method signatures of subservice API methods.

**Location:** `http://smile.example.com:port/stage/smile/ws/v2/subservice`

**WSDL:** `http://smile.example.com:port/stage/smile/ws/v2/subservice?wsdl`

**Table 23: Plan change API methods**

| Method | Method signature | Purpose |
|---|---|---|
| addSubservice() | `Subservice addSubservice(`<br>`string usn,`<br>`SubserviceRequest subservice)` | Creates a new subservice associated to an existing subscription. |
| addSubservices() | `addSubservices(`<br>`string usn,`<br>`SubserviceRequestList subserviceList)` | Creates bulk subservices associated with the same subscription. |
| getSubservice() | `getSubservice(`<br>`string ssid)` | Returns a single subservice, identified by SSID. |
| getSubserviceBySubserviceUsername() | `getSubserviceBySubserviceUsername(`<br>`int sid,`<br>`string subserviceUsername)` | Returns a subservice that is linked to a specified service, identified by unique service (SID) and subservice username. |
| getSubserviceForSubscription() | `getSubservicesForSubscription(`<br>`string usn)` | Returns a list of subservices for a specified subscription. |
| removeSubservice() | `removeSubservice(`<br>`string ssid)` | Deletes a subservice. |
| updateSebservice() | `updateSubservice(`<br>`Subservice sNew)` | Alters information of a subservice. |

Related concepts

Subservice API overview on page 232

# Appendix B

# Message bus topics

## SmileAccount

Smile emits messages in SmileAccount to notify listeners of account changes.

The messages in SmileAccount are:

- `AccountUpdated`

### `AccountUpdated`

Smile emits this message when an account is updated.

#### Message header

The following table lists the parameters in the AccountUpdated message header.

**Table 24: AccountUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| accountUsn | string | The unique ID for the account |
| company | integer | The company ID of the account |
| smileMessageType | string | "AccountUpdated" |

#### Message body

The message body contains the `<Account>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Account
```

#### Example: The `<Account>` element

This example shows an Account XML document that contains the `<Account>` element.

```
<Account xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142421540</USN>
  <AccountType key="1">Standard Account</AccountType>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
```

```
    <CycleDay>31</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>31</CycleDay>
  </RatingCycle>
  <Timezone key="13">Australia/Melbourne</Timezone>
  <AccountTerms key="326">Exception terms</AccountTerms>
  <Properties>
    <Object>
      <String name="abn"/>
      <String name="billAddress"/>
      <String name="billBuildingName"/>
      <String name="billCountry"/>
      <String name="billFloorNumber"/>
      <String name="billLotNumber"/>
      <String name="billName"/>
      <String name="billNumber"/>
      <String name="billPostCode"/>
      <String name="billPostalDeliveryType"/>
      <String name="billState"/>
      <String name="billSubUnit"/>
      <String name="billSuburb"/>
      <String name="billType"/>
      <String name="comments"/>
      <String name="companyPosition"/>
      <String name="contactFamily"/>
      <String name="contactGiven"/>
      <String name="contactTitle"/>
      <String name="emailAddress"/>
      <String name="fax"/>
      <String name="homePhone"/>
      <String name="jabberAddress"/>
      <String name="mobilePhone"/>
      <String name="name"/>
      <String name="secretAnswer"/>
      <String name="secretQuestion"/>
      <String name="sex"/>
      <String name="streetAddress"/>
      <String name="streetBuildingName"/>
      <String name="streetCountry"/>
      <String name="streetFloorNumber"/>
      <String name="streetLotNumber"/>
      <String name="streetName"/>
      <String name="streetNumber"/>
      <String name="streetPostalDeliveryType"/>
      <String name="streetPostcode"/>
      <String name="streetState"/>
      <String name="streetSubUnit"/>
      <String name="streetSuburb"/>
      <String name="streetType"/>
      <String name="tradingName"/>
      <String name="workPhone"/>
      <String name="username">2142421540</String>
      <String name="alternateAccountNumber"/>
      <Integer name="preferredHomeContactTime"/>
      <Integer name="preferredMobileContactTime"/>
```

```
      <Integer name="preferredWorkContactTime"/>
      <Boolean name="billPrintName">false</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
      <Timestamp name="dob">2019-08-02T00:00:00.000+10:00</Timestamp>
    </Object>
  </Properties>
</Account>
```

## SmileCharge

Smile emits messages in SmileCharge whenever a deferred charge (a charge that is raised against an account, but not assigned to an invoice until the end of the current billing period) is created or updated.

The messages in SmileCharge are:

- `ChargeDeferred`

### `ChargeDeferred`

Smile emits this message when a deferred charge is created or updated.

#### Message header

The following table lists the parameters in the ChargeDeferred message header.

**Table 25: ChargeDeferred header parameters**

| Parameter | Type | Values |
|---|---|---|
| smileMessageType | string | "ChargeDeferred" |
| company | integer | Company that the account holder that the charge is being raised against belongs to |
| accountType | integer | Numeric descriptor for the account type. This can also be obtained through the SOAP API method `WSAccount.get()` |
| usn | string | USN identifying the account that the charge is being raised against |
| chargeId | UUID | Identifies the charge itself |
| currency | string | 3-letter ISO 4217 currency code indicating what currency the charge has been issued in |
| amount | decimal | Amount of the charge |

#### Message body

The root element of the message body is `<RealisedCharge>` and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}RealisedCharge
```

#### Example: The `<RealisedCharge` element

This example shows an RealisedCharge XML document that contains the `<RealisedCharge>` element.

```
<RealisedCharge xmlns="http://xml.inomial.com/smile/2.xsd">
  <ChargeID>0f11aa6a-476a-47e4-87ed-cb4d828579e8</ChargeID>
  <ItemDescriptionID>3gwifimodem</ItemDescriptionID>
  <Description>3G Wireless Modem</Description>
  <ChargeStart>2016-01-15T00:00:00.000+11:00</ChargeStart>
  <ChargeEnd>2016-01-15T00:00:00.000+11:00</ChargeEnd>
  <Quantity>1</Quantity>
```

```
  <Count>1</Count>
  <Charge>49.95</Charge>
  <ChargeGST>true</ChargeGST>
  <GSTAmount>4.99</GSTAmount>
  <USN>2142426622</USN>
  <RealisingTransactionItems>
    <TransactionItem>
      <usn>2142426622</usn>
      <lineNumber>0</lineNumber>
      <itemCode>3gwifimodem</itemCode>
      <chargeFrom>2016-01-15+11:00</chargeFrom>
      <chargeTo>2016-01-15+11:00</chargeTo>
      <description>3G Wireless Modem</description>
      <quantity>1</quantity>
      <currency>AUD</currency>
      <amount>49.95</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0</discount>
      <count>1</count>
      <gstAmount>4.99</gstAmount>
      <salesReportCategory>invoice</salesReportCategory>
      <excludeFromInvoiceGrouping>false</excludeFromInvoiceGrouping>
      <Object objectId="6118" formId="978" displayName="Name: Example
 Internet  Colour: Turquoise">
        <String name="companyName">Example Internet</String>
        <String name="colour">Turquoise</String>
      </Object>
      <chargeId>0f11aa6a-476a-47e4-87ed-cb4d828579e8</chargeId>
    </TransactionItem>
  </RealisingTransactionItems>
</RealisedCharge>
```

## SmileConnection

Smile emits messages in the SmileConnection to notify listeners about RADUIS connection changes.

The messages in SmileConnection are:

- `ConnectionFeaturesUpdated`

### `ConnectionFeaturesUpdated`

Smile emits this message when:

- features are applied to connections automatically or manually
- a RADIUS data session on the current online connection via Smile is disconnected

#### Message header

The following table lists the parameters in the ConnectionFeaturesUpdated message header.

**Table 26: SubscriptionFeaturesChanged header parameters**

| Parameter | Type | Values |
|---|---|---|
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| usn | string | The unique subscriber number for the subscription the order is raised against |
| sid | integer | The service ID |
| nasID | integer | The ID of the Smile NAS Configuration that the connection is through |
| smileMessageType | string | "ConnectionFeaturesUpdated" |

#### Message body

The message body contains the `<ActiveConnection>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}ActiveConnection
```

#### Example: The `<ActiveConnection>` element

This example shows an ActiveConnection XML document that contains the `<ActiveConnection>` element.

```
<ActiveConnection xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142427877</USN>
  <NasId>516</NasId>
  <ActiveFeatures>
    <Feature>
      <FeatureId>600</FeatureId>
      <Name>Test Feature 600</Name>
      <Object/>
    </Feature>
```

```
  </ActiveFeatures>
  <IpAddress>10.10.10.209</IpAddress>
  <Username>radius@inomial.com</Username>
  <AccountSessionId>SID:1425606499964:0</AccountSessionId>
  <Caller>1800123456</Caller>
  <Called>1800111111</Called>
  <SessionId>553</SessionId>
</ActiveConnection>
```

### Related reference

applyFeatureToConnections() on page 182

## SmileCustomCollection

Smile emits messages in SmileCustomCollection whenever a custom collection entry is created, updated or deleted.

Note that currently there are no messages on whether a custom collection itself is created, updated or deleted - these operations have to be done via the Smile configuration UI itself, and external third-party applications must be manually made aware of the existence of your custom collection databases.

All messages on this topic have a common header.

The messages in SmileCustomCollection are:

- `CustomCollectionEntryAdded`
- `CustomCollectionEntryUpdated`
- `CustomCollectionEntryDeleted`

### `CustomCollectionEntryAdded`

Smile emits this message when a new collection entry is created.

#### Message header

The following table lists the parameters in the CustomCollectionEntryAdded message header.

**Table 27: CustomCollectionEntryAdded header parameters**

| Parameter | Type | Values |
|---|---|---|
| companyId | integer | Company that the custom collection entry belongs to |
| key | integer | Unique custom collection entry key that identifies the entry being added |
| smileMessageType | string | "CustomCollectionEntryAdded" |
| collectionCode | string | Code name of the collection possessing the entry that this message concerns |
| entryCode | string | Code name of the collection entry that this message concerns |

#### Message body

The root element of the message body is `<CustomCollectionEntry>` and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}CustomCollectionEntry
```

The message body always reflects the latest (post-operation) state of the custom collection entry.

It should be noted that the objectId="..." attribute of the `<Object>` element is purely intended for Inomial's own diagnostic reference, and is not intended to be used by third-party developers.

If the values of the objectId="..." and top-level key="..." attributes appear identical, this is purely of coincidence, and merely a side-effect of the underlying storage system used in Smile. You should not rely

on the value of the objectId="..." attribute as Inomial reserves the right to alter the way this attribute is generated in future versions of Smile without notice.

### Example: The `<CustomCollectionEntry` element

This example shows an CustomCollectionEntry XML document that contains the `<CustomCollectionEntry>` element.

```
<CustomCollectionEntry key="6564" xmlns="http://xml.inomial.com/
smile/2.xsd">
  <CollectionCode>products</CollectionCode>
  <EntryCode>MobilePhoneModel</EntryCode>
  <Object objectId="6564" formId="1156" displayName="iPhone 6S
 (Silver)">
    <String name="productCode">iphone6s</String>
    <String name="productDescription">iPhone 6S</String>
    <String name="colour">Silver</String>
  </Object>
</CustomCollectionEntry>
```

Related reference

addCollectionEntry() on page 355

## `CustomCollectionEntryUpdated`

Smile emits this message when an existing collection entry is modified.

### Message header

The following table lists the parameters in the CustomCollectionEntryUpdated message header.

**Table 28: CustomCollectionEntryUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| companyId | integer | Company that the custom collection entry belongs to |
| key | integer | Unique custom collection entry key that identifies the entry being updated |
| smileMessageType | string | "CustomCollectionEntryUpdated" |
| collectionCode | string | Code name of the collection possessing the entry that this message concerns |
| entryCode | string | Code name of the collection entry that this message concerns |

### Message body

The root element of the message body is `<CustomCollectionEntry>` and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}CustomCollectionEntry
```

The message body always reflects the latest (post-operation) state of the custom collection entry.

In the event of a custom collection entry's `entryCode` being renamed via `updateCollectionEntry()`, both the `entryCode` header property and <EntryCode> element will contain the new name, but the key will always remain the same, allowing you to identify which collection entry has been renamed.

It should be noted that the objectId="..." attribute of the `<Object>` element is purely intended for Inomial's own diagnostic reference, and is not intended to be used by third-party developers.

If the values of the objectId="..." and top-level key="..." attributes appear identical, this is purely of coincidence, and merely a side-effect of the underlying storage system used in Smile. You should not rely the value of the objectId="..." attribute as Inomial reserves the right to alter the way this attribute is generated in future versions of Smile without notice.

### Example: The `<CustomCollectionEntry` element

This example shows an CustomCollectionEntry XML document that contains the `<CustomCollectionEntry>` element.

```
<CustomCollectionEntry key="6564" xmlns="http://xml.inomial.com/
smile/2.xsd">
  <CollectionCode>products</CollectionCode>
  <EntryCode>MobilePhoneModel</EntryCode>
  <Object objectId="6564" formId="1156" displayName="iPhone 6S
 (Silver)">
    <String name="productCode">iphone6s</String>
    <String name="productDescription">iPhone 6S</String>
    <String name="colour">Silver</String>
  </Object>
</CustomCollectionEntry>
```

### Related reference

updateCollectionEntry() on page 357

### `CustomCollectionEntryDeleted`

Smile emits this message when an existing collection entry is deleted.

#### Message header

The following table lists the parameters in the CustomCollectionEntryDeleted message header.

**Table 29: CustomCollectionEntryDeleted header parameters**

| Parameter | Type | Values |
|---|---|---|
| companyId | integer | Company that the custom collection entry belongs to |
| key | integer | Unique custom collection entry key that identifies the entry being deleted |
| smileMessageType | string | "CustomCollectionEntryDeleted" |
| collectionCode | string | Code name of the collection possessing the entry that this message concerns |
| entryCode | string | Code name of the collection entry that this message concerns |

#### Message body

CustomCollectionEntryDeleted messages does not have a message body. Only the message header properties are sent.

Related reference

deleteCollectionEntry() on page 359

## SmileTicket

Smile emits messages in SmileTicket to notify listeners that communication with a customer has occurred, for example, a notification was sent or a ticket was updated.

The messages in SmileTicket are:

- `CorrespondenceSent`

Related concepts

CRM API overview on page 360

### `CorrespondenceSent`

Smile emits this message after a notification is sent to a customer about a spend limit, credit limit, usage alert group or value pool. The notification is triggered when a subscription crosses an alert threshold, or a new rating period begins and the spend management tools are reset to 0.

This message does not confirm that the customer received the correspondence. If the notification is not successfully sent, Smile does not emit any SmileTicket message.

#### Message header

The following table lists the parameters in the CorrespondenceSent message header.

**Table 30: CorrespondenceSent header parameters**

| Parameter | Type | Values |
|---|---|---|
| company | integer | The company ID this account belongs to |
| accountUsn | string | The USN of the account that triggered the notification |
| subscriptionUsn | string | The USN of the subscription that triggered the notification |
| notificationType | uuid | The identifier of the notification type |
| channel | string | "SMS", "Email" |
| smileMessageType | string | "CorrespondenceSent" |

#### Message body

The message body contains the `<TransmittedCorrespondence>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransmittedCorrespondence
```

#### Example: The `<TransmittedCorrespondence>` element for an SMS notification

This example shows a TransmittedCorrespondence XML document for an SMS notification.

```
<TransmittedCorrespondence xmlns="http://xml.inomial.com/
smile/2.xsd">
  <company>1</company>
```

```
  <accountUsn>2142422670</accountUsn>
  <subscriptionUsn>2142425376</subscriptionUsn>
  <notificationType>
    <notificationTypeId>a32b46cf-b6af-50e8-bbff-c69b80c722df</
notificationTypeId>
    <name>Included Calls</name>
  </notificationType>
  <channel>SMS</channel>
  <transmissionTime>2012-10-19T15:55:05.847+11:00</transmissionTime>

 <receiptId>&lt;33299522.6.1350622505767.JavaMail.joe@example.lan&gt;</
receiptId>
  <operatorUsn>2142420716</operatorUsn>
  <SMS>
    <RecipientList>
      <address>joe@example.com</address>
    </RecipientList>
    <from>admin@example.com</from>
    <bodyText>You've used 100% of your $10.00 Included Calls
 allowance
      0491570156</bodyText>
  </SMS>
</TransmittedCorrespondence
```

### Example: The `<TransmittedCorrespondence>` element for an email notification

This example shows a TransmittedCorrespondence XML document for an email notification.

```
<TransmittedCorrespondence xmlns="http://xml.inomial.com/
smile/2.xsd">
  <company>1</company>
  <accountUsn>2142422670</accountUsn>
  <subscriptionUsn>2142425376</subscriptionUsn>
  <notificationType>
    <notificationTypeId>a32b46cf-b6af-50e8-bbff-c69b80c722df</
notificationTypeId>
    <name>Included Calls</name>
  </notificationType>
  <channel>Email</channel>
  <transmissionTime>2012-10-19T16:15:52.647+11:00</transmissionTime>

 <receiptId>&lt;22230889.13.1350623750015.JavaMail.joe@example.lan&gt;</
receiptId>
  <operatorUsn>2142420716</operatorUsn>
  <Email>
    <RecipientList>
      <address>joe@example.com</address>
    </RecipientList>
    <CCList/>
    <BCCList/>
    <from>admin@example.com</from>
    <replyToList/>
    <subject>Value Pool consumption</subject>
    <bodyText>% Consumed 100
      Alert Threshold % 100
      Amount Consumed 10.00
```

```
      Company Name Example Telecom Ltd
      Contact Display Name Joe
      Contact Family Name
      Contact Given Name
      Contact Name
      Current Billing Period End 31/10/12
      Current Plan Code ABC-A2
      Current Plan Name A2 plan
      Current Time 16:15:49
      Formatted Amount Consumed $10.00
      Next Billing Period Start 1/11/12
      Raw Username 0415050111
      Sequence Number 10004
      Subscription Description
      Today's Date 19/10/12
      Today's Date and Time 16:15 19/10/2012
      User Service Name Road Warrior (0491570156)
      Value Pool Amount 10.00
      Value Pool Formatted Amount $10.00
      Value Pool Name Included Calls
      Reset Timestamp 22:00 31/10/2012
      Reset Timestamp (customer timezone) 00:00 01/11/2012
      Account Contact Display Name Joe
      Account Contact Family Name Smith
      Account Contact Given Name Joe
      Account Contact Name Smith
      Account Company Name
      Subscription Company Name


      ---13635400---
    </bodyText>
    <bodyContentType>text/plain</bodyContentType>
    <Attachments/>
  </Email>
</TransmittedCorrespondence>
```

## SmileNumberPool

Smile emits messages in the SmileNumberPool to notify listeners of pool reservation changes.

The messages in SmileNumberPool are:

- `LinkDeallocation`

Related concepts

Manage number pools on page 411

### `LinkDeallocation`

Smile emits this message after a number pool reservation is deallocated.

#### Message header

The following table lists the parameters in the LinkDeallocation message header.

**Table 31: LinkDeallocation header parameters**

| Parameter | Type | Values |
|---|---|---|
| linkId | string | The ID of the link which has been deallocated |
| quarantineLinkId | string | The ID of the quarantine link created due to the deallocation (if one was created) |
| reservationStatus | boolean | The status of the reservation if it still has active links and continues to reserve the resources |
| smileMessageType | string | "LinkDeallocation" |

#### Message body

The message body contains the `<PoolReservationLink>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}PoolReservationLink
```

#### Example: The `<PoolReservationLink>` element

This example shows a PoolReservationLink XML document that contains the <PoolReservationLink> element, describing a link which has been unlinked.

```
<PoolReservationLink xmlns="http://xml.inomial.com/smile/2.xsd"
 xmlns="http://ws.inomial.com/smile.2">
  <LinkId>e55fe410-5d6f-43f3-bbf1-51bafb2f07fe</LinkId>
  <LinkType>Allocation</LinkType>
  <Reservation>
    <ReservationId>732b30ff-824b-4a0e-9eea-691e280f2d5e</
ReservationId>
    <Start>0399990000</Start>
    <End>0399999999</End>
  </Reservation>
```

```
</PoolReservationLink>
```

### Related reference

unlink() on page 418

## SmileOrder

Smile emits messages in the SmileOrder to notify listeners when an order has completed a stage in the order process.

The messages in SmileOrder are:

- `OrderApproved`
- `OrderCancelled`
- `OrderComplete`
- `OrderEntered`
- `OrderRequiresApproval`

Related concepts

Order API overview on page 37

### `OrderApproved`

Smile emits this message when an order is approved. The order then proceeds to the next stage of the order process, for example, completion or provisioning.

Smile automatically approves orders that do not require approval.

#### Message header

The following table lists the parameters in the OrderApproved message header.

**Table 32: OrderApproved header parameters**

| Parameter | Type | Values |
|---|---|---|
| number | string | The unique order number that identifies the order |
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| smileMessageType | string | "OrderApproved" |

#### Message body

The message body contains the `<Order>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Order
```

#### Example: The `<Order>` element

This example shows an Order XML document that contains the `<Order>` element.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>false</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
```

```
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
    </Object>
  </OrderForm>
</Order>
```

Related concepts

Message sequence—Account orders on page 72

Message sequence—Subscription orders on page 73

Related reference

approveOrder() on page 53

## OrderCancelled

Smile emits this message when an order has been cancelled and will no longer proceed.

### Message header

The following table lists the parameters in the OrderCancelled message header.

**Table 33: OrderCancelled header parameters**

| Parameter | Type | Values |
|---|---|---|
| number | string | The unique order number that identifies the order |
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| smileMessageType | string | "OrderCancelled" |

### Message body

The message body contains the `<Order>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Order
```

### Example: The `<Order>` element

This example shows an Order XML document that contains the `<Order>` element.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>false</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
```

```
    </Object>
  </OrderForm>
</Order>
```

### OrderComplete

Smile emits this message when an order is complete. The order is approved and any subscriptions in the order are provisioned, enabled and activated.

#### Message header

The following table lists the parameters in the OrderComplete message header.

**Table 34: OrderComplete header parameters**

| Parameter | Type | Values |
|---|---|---|
| number | string | The unique order number that identifies the order |
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| smileMessageType | string | "OrderComplete" |

#### Message body

The message body contains the `<Order>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Order
```

#### Example: The `<Order>` element

This example shows an Order XML document that contains the `<Order>` element.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>false</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
    </Object>
  </OrderForm>
</Order>
```

#### Related concepts

Message sequence—Account orders on page 72
Message sequence—Subscription orders on page 73

## OrderEntered

Smile emits this message when an order is entered. The order is now pending approval by an operator or pending automatic approval by Smile.

### Message header

The following table lists the parameters in the OrderEntered message header.

**Table 35: OrderEntered header parameters**

| Parameter | Type | Values |
|---|---|---|
| number | string | The unique order number that identifies the order |
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| smileMessageType | string | "OrderEntered" |

### Message body

The message body contains the `<Order>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Order
```

### Example: The `<Order>` element

This example shows an Order XML document that contains the `<Order>` element. Note that the order requires approval.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>true</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
    </Object>
  </OrderForm>
</Order>
```

Related concepts

Message sequence—Account orders on page 72
Message sequence—Subscription orders on page 73

Related reference

orderAccount() on page 46
orderSubscription() on page 48
placeOrderGroup() on page 62

### `OrderRequiresApproval`

Smile emits this message when an order must be approved before it can proceed to the next stage in the order process. Applications can listen for this message to alert operators or other processes that an order is pending approval.

#### Message header

The following table lists the parameters in the OrderRequiresApproval message header.

**Table 36: OrderRequiresApproval header parameters**

| Parameter | Type | Values |
|---|---|---|
| number | string | The unique order number that identifies the order |
| accountUsn | string | The unique subscriber number for the account the order is raised against |
| smileMessageType | string | "OrderRequiresApproval" |

#### Message body

The message body contains the `<Order>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Order
```

#### Example: The `<Order>` element

This example shows an Order XML document that contains the `<Order>` element. Note that the order requires approval.

```
<Order xmlns="http://xml.inomial.com/smile/2.xsd">
  <Package>192</Package>
  <RequiresApproval>true</RequiresApproval>
  <AccountUSN>1204921324</AccountUSN>
  <PackagePlan>812</PackagePlan>
  <OrderForm>
    <Object>
      <String name="username">+61355501256</String>
      <String name="contactGiven">Joe</String>
      <String name="contactFamily">Smith</String>
      <Integer name="phoneLines">12</Integer>
    </Object>
  </OrderForm>
</Order>
```

#### Related concepts

Message sequence—Account orders on page 72

Message sequence—Subscription orders on page 73

## SmileSubscription

Smile emits messages in SmileSubscription to notify listeners of subscription changes.

The messages in SmileSubscription are:

- `CreditLimitExceeded`
- `CreditLimitOkay`
- `PrepaidExpiryWarningEvent`
- `RatingConfigurationUpdated`
- `SubscriptionActivated`
- `SubscriptionClosed`
- `SubscriptionEnabled`
- `SubscriptionFeaturesChanged`
- `SubscriptionIpRangesUpdated`
- `SubscriptionPlanChanged`
- `SubscriptionProvisionCommand`
- `SubscriptionProvisioned`
- `SubscriptionProvisionResponse`
- `SubscriptionReadyToProvision`
- `SubscriptionResumed`
- `SubscriptionSpendLimitExceeded`
- `SubscriptionSpendLimitReceded`
- `SubscriptionSpendLimitStatusUpdated`
- `SubscriptionSuspended`
- `SubscriptionUpdated`

### Related concepts

Subscription API overview on page 156
Listen for spend limit changes on page 280

### `CreditLimitExceeded`

Smile emits this message when a subscription's usage causes the account to exceed the subscription credit limit. When calculating a subscription credit limit, Smile includes the value of outstanding account balances and any usage in the current invoicing cycle for that subscription. However, the subscription credit limit does not include the value of any usage in the current invoicing cycle for other subscriptions in the account.

#### Message header

The following table lists the parameters in the CreditLimitExceeded message header.

**Table 37: CreditLimitExceeded header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| creditLimit | string | The credit limit for the subscription or account |
| balance | string | The balance of the account associated with the subscription |
| currency | string | The ISO 4217 currency code of the account associated with the subscription |
| smileMessageType | string | "CreditLimitExceeded" |

### Message body

The message body contains the `<Subscription>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Subscription
```

### Example: The `<Subscription>` element

This example shows a Subscription XML document that contains the `<Subscription>` element.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142426838</USN>
  <SID>286</SID>
  <ServiceName>Test service 286</ServiceName>
  <Parent>2142426820</Parent>
  <Properties>
    <Object>
      <String name="username">joe.smith@example.com</String>
      <Boolean name="billPrintName">false</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <CostCentre key="1">Example cost centre</CostCentre>
</Subscription>
```

### `CreditLimitOkay`

Smile emits this message when an account that had exceeded its credit limit is within the limit again.

#### Message header

The following table lists the parameters in the CreditLimitOkay message header.

**Table 38: CreditLimitOkay header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| creditLimit | string | The credit limit for the subscription or account |
| balance | string | The balance of the account associated with the subscription |
| currency | string | The ISO 4217 currency code of the account associated with the subscription |
| smileMessageType | string | "CreditLimitOkay" |

#### Message body

The message body contains the `<Subscription>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}Subscription
```

#### Example: The `<Subscription>` element

This example shows a Subscription XML document that contains the `<Subscription>` element.

```
<Subscription xmlns="http://xml.inomial.com/smile/2.xsd">
  <USN>2142426838</USN>
  <SID>286</SID>
  <ServiceName>Test service 286</ServiceName>
  <Parent>2142426820</Parent>
  <Properties>
    <Object>
      <String name="username">joe.smith@example.com</String>
      <Boolean name="billPrintName">false</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  <InvoicingCycle>
    <CycleType>Anniversary</CycleType>
    <CycleDay>24</CycleDay>
  </InvoicingCycle>
  <RatingCycle>
    <CycleType>Anniversary</CycleType>
```

```
    <CycleDay>24</CycleDay>
  </RatingCycle>
  <Timezone key="19">Australia/Sydney</Timezone>
  <CostCentre key="1">Example cost centre</CostCentre>
</Subscription>
```

## PrepaidExpiryWarningEvent

Smile emits this message when a usage block for a subscription is due to expire in a certain amount of time.

The advance intervals of time when warnings are issued are set by creating a prepaid warning threshold for a given prepaid block under **Prepaid Time and Data** in Smile. For more information, see the *Configuration Guide*.

### Message header

The following table lists the parameters in the PrepaidExpiryWarningEvent message header.

**Table 39: PrepaidExpiryWarningEvent header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "PrepaidExpiryWarningEvent" |

### Message body

The message body contains the `<PrepaidExpiryWarningEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}PrepaidExpiryWarningEvent
```

### Example: The `<PrepaidExpiryWarningEvent>` element

This example shows a PrepaidExpiryWarningEvent XML document for a message issued 24 hours before this usage block is due to expire.

The subscriber purchased a usage block, with prepaid ID 34359, on 4th June 2015 at 2:56:54pm. The block provided 6 hours of online time and would expire after 30 days. At the time of the event message the subscriber has consumed 2 hours and 25 minutes of online time from their total quota of 6 hours.

Although the time-to-expiry and quantity consumed are not present in the message, they can be derived by the following formula:

*time_to_expiry = EndDate - EventTimestamp*

*quantity_consumed = PurchasedQuantity - RemainingQuantity*

```
<PrepaidExpiryWarningEvent xmlns="http://xml.inomial.com/
smile/2.xsd">
```

```
  <Subscription>
    <USN>2142423785</USN>
    <SID>770</SID><
  </Subscription>
  <PrepaidWarningLogId>13413</PrepaidWarningLogId>
  <EventTimestamp>2015-07-03T14:56:54.231+10:00</EventTimestamp>
  <PrepaidState>
    <PrepaidId>34359</PrepaidId>
    <StartDate>2015-06-04T14:56:54.231+10:00</StartDate>
    <EndDate>2015-07-04T14:56:54.231+10:00</EndDate>
    <PrepaidDescription key="beanctr">Bean Counter Pack</
PrepaidDescription>
    <PurchasedQuantity formatted="0:06:00">360</PurchasedQuantity>
    <RemainingQuantity formatted="0:03:35">215</RemainingQuantity>
    <ChargeType key="0">Online time</ChargeType>
  </PrepaidState>
</PrepaidExpiryWarningEvent>
```

### `RatingConfigurationUpdated`

RatingConfigurationUpdated messages will trigger a re-rate when configuration for the rating is changed.

Smile emits this message when one of the following occurs:

- `updateSubscriptionContract()` is called
- a subscription feature is added when `applyFeature()` is called
- a subscription feature is updated when `stopApplyingFeature()` is called
- Smile receives a `SubscriptionSpendLimitExceeded` or `SubscriptionSpendLimitReceded` message
- `addSimpleRecurringCharges()` or `updateSimpleRecurringCharges()` is called
- `resume()`, `suspend()` or `Subscription.changePlan()` is called
- `changeManagedPlanGroup()` or `Plan.changePlan()` is called
- `commitPlanChange()` or `proposePlanChange()` is called
- prepaid time is purchased via CAT
- a document action is executed to enable or disable a service feature on all "auto lock" subscriptions on an account
- prepaid is added, removed or saved
- an adjustment in prepaid block usage is added or saved
- a prepaid time block is removed
- a recurring charge is added or saved
- a RADIUS feature is added, removed or saved
- a plan schedule item is added, removed or saved
- a voice service is re-rated
- a destination in voice service is added, removed or saved
- process a returned provision request when it requests to provision or change plan

#### Message header

The following table lists the parameters in the RatingConfigurationUpdated message header.

**Table 40: RatingConfigurationUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| usn | string | The unique subscription ID number for the subscription |
| sid | integer | The service ID |
| ratingPriorityState | integer | Specifies the priority of a re-rate, "1" (priority) or "0" (normal) |
| timestamp | long | (Optional) The date and time from when the change will be recognised |
| smileMessageType | string | "RatingConfigurationUpdated" |

### Message body

This message type does not have a message body.

#### Related reference

suspend() on page 177
resume() on page 179
applyFeature() on page 180
stopApplyingFeature() on page 183
Subscription.changePlan() on page 193
proposePlanChange() on page 200
commitPlanChange() on page 204
Plan.changePlan() on page 256
changeManagedPlanGroup() on page 260
updateSubscriptionContract() on page 342
updateSimpleRecurringCharges() on page 395
addSimpleRecurringCharges() on page 396

## `SubscriptionActivated`

Smile emits this message after billing is activated for a new subscription. Activation may raise a setup fee against the subscription and start the subscription's plan and recurring charges.

A subscription must be provisioned, enabled and activated before it is fully available. These three events can happen in any order. For example, you may wish to provision and enable a subscription immediately, but delay activation until a piece of hardware such as a modem or mobile handset reaches the customer.

### Message header

The following table lists the parameters in the SubscriptionActivated message header.

**Table 41: SubscriptionActivated header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |

| Parameter | Type | Values |
|---|---|---|
| smileMessageType | string | "SubscriptionActivated" |

### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned, enabled and activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

## Related concepts

Message sequence—Subscription orders on page 73

### SubscriptionClosed

Smile emits this message after a subscription is closed and billing is disabled. Business intelligence and audit listeners may use this information for reporting purposes. Listeners in other systems may need to disable or invalidate a user after receiving this message.

#### Message header

The following table lists the parameters in the SubscriptionClosed message header.

**Table 42: SubscriptionClosed header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionClosed" |

#### Message body

The message body contains the `<SubscriptionUpdated>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionUpdated
```

#### Example: The `<SubscriptionUpdated>` element

This example shows a SubscriptionUpdated XML document that contains the `<SubscriptionUpdated>` element for a closed subscription. Note that the subscription is deprovisioned and no longer activated.

```
<SubscriptionUpdated xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
    <USN>2142426697</USN>
    <SID>17</SID>
    <ServiceName>Smile Access</ServiceName>
    <Parent>2142426689</Parent>
    <Properties>
      <Object>
        <String name="emailAddress">joe@example.com</String>
        <String name="username">joe@example.com</String>
      </Object>
    </Properties>
    <ProvisionStatus>Deprovisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
</SubscriptionUpdated>
```

Related reference

cancel() on page 176

### SubscriptionEnabled

Smile emits this message after a new subscription is enabled and authorised to access the network.

A subscription must be provisioned, enabled and activated before it is fully available. These three events can happen in any order. For example, you may wish to provision and activate a subscription immediately, but delay enablement until after you receive an initial payment from the customer.

#### Message header

The following table lists the parameters in the SubscriptionEnabled message header.

**Table 43: SubscriptionEnabled header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionEnabled" |

#### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

#### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned and enabled, but is not yet activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
```

```
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

### Related concepts

Message sequence—Subscription orders on page 73

## SubscriptionFeaturesChanged

Smile emits this message when a subscription's features have changed.

This message contains the current set of active features for the subscription but does not specify which features have changed or been removed. To determine which features have changed, listeners must persist their own set of active features and compare this set to the set in the message.

### Message header

The following table lists the parameters in the SubscriptionFeaturesChanged message header.

**Table 44: SubscriptionFeaturesChanged header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionFeaturesChanged" |

### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned, enabled and activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
```

```
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

Related reference

applyFeature() on page 180
stopApplyingFeature() on page 183
Subscription.changePlan() on page 193
redeemVoucher() on page 315
addPrepaid() on page 318
addSelfServePrepaid() on page 327

## `SubscriptionIpRangesUpdated`

Smile emits this message when the IP addresses associated with the subscription change. In particular when one of the following occurs:

- `addIpRange()`, `addIpRangeReservation()`, `removeIpRange()` or `removePoolReservationLink()` is called
- a pool reservation is removed when Smile receives a `LinkDeallocation` message
- all reservation links are removed when Smile receives a `SubscriptionClosed` message
- a pool reservation link is removed

### Message header

The following table lists the parameters in the SubscriptionIpRangesUpdated message header.

**Table 45: SubscriptionIpRangesUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionIpRangesUpdated" |

### Message body

The message body contains the `<SubscriptionIpRangesUpdated>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionIpRangesUpdated
```

### Example: The `<SubscriptionIpRangesUpdated>` element

This example shows a SubscriptionIpRangesUpdated XML document for a subscription that has 4 ranges of IP addresses allocated to it: one from an IP Pool reservation, one IPv6 range, and 2 IPv4 ranges that are not managed by an IP Pool.

```
<SubscriptionIpRangesUpdated xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142425772</USN>
    <SID>725</SID>
    <ServiceName>Test service 725</ServiceName>
    <Parent>2142425764</Parent>
    <Properties>
      <Object>
        <String name="abn"/>
        <String name="billAddress"/>
        <String name="billBuildingName"/>
        <String name="billCountry"/>
        <String name="billFloorNumber"/>
        <String name="billLotNumber"/>
        <String name="billName"/>
        <String name="billNumber"/>
        <String name="billPostCode"/>
        <String name="billPostalDeliveryType"/>
        <String name="billState"/>
        <String name="billSubUnit"/>
        <String name="billSuburb"/>
        <String name="billType"/>
        <String name="comments"/>
        <String name="companyPosition"/>
        <String name="contactFamily"/>
        <String name="contactGiven"/>
        <String name="contactTitle"/>
        <String name="emailAddress"/>
        <String name="fax"/>
        <String name="homePhone"/>
        <String name="jabberAddress"/>
        <String name="mobilePhone"/>
        <String name="name"/>
        <String name="secretAnswer"/>
        <String name="secretQuestion"/>
        <String name="sex"/>
        <String name="streetAddress"/>
        <String name="streetBuildingName"/>
        <String name="streetCountry"/>
        <String name="streetFloorNumber"/>
        <String name="streetLotNumber"/>
        <String name="streetName"/>
```

```
        <String name="streetNumber"/>
        <String name="streetPostalDeliveryType"/>
        <String name="streetPostcode"/>
        <String name="streetState"/>
        <String name="streetSubUnit"/>
        <String name="streetSuburb"/>
        <String name="streetType"/>
        <String name="tradingName"/>
        <String name="workPhone"/>
        <String name="username">test1422428588258@example.com</
String>
        <String name="description"/>
        <String name="filterId"/>
        <String name="framedInterfaceId"/>
        <String name="framedIpAddress"/>
        <String name="framedIpNetmask"/>
        <String name="framedIpv6Pool"/>
        <String name="framedIpv6Route"/>
        <String name="framedNetworkPrefix"/>
        <String name="framedRoute"/>
        <Integer name="preferredHomeContactTime"/>
        <Integer name="preferredMobileContactTime"/>
        <Integer name="preferredWorkContactTime"/>
        <Integer name="framedProtocol"/>
        <Integer name="radiusServiceType"/>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
        <Timestamp name="dob"/>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <InvoicingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>31</CycleDay>
    </InvoicingCycle>
    <RatingCycle>
      <CycleType>Anniversary</CycleType>
      <CycleDay>31</CycleDay>
    </RatingCycle>
    <Timezone key="13">Australia/Melbourne</Timezone>
    <Company key="1">Example Internet</Company>
  </Subscription>
  <IPRanges>
    <SubscriptionIpRange>
      <SubscriptionIpRangeId>46799d37-4fde-4fb0-8450-060dd64b5666</
SubscriptionIpRangeId>
      <FromAddress>127.0.0.6</FromAddress>
      <ToAddress>127.0.0.10</ToAddress>
      <PoolReservationLink>
        <LinkId>3bcb7026-6143-4507-88da-3bf656d3a9fe</LinkId>
        <LinkType>Allocation</LinkType>
        <Reservation>
          <ReservationId>dc82d419-5ffd-4bcf-9b73-4b59957d1590</
ReservationId>
```

```
          <Start>127.0.0.6</Start>
          <End>127.0.0.10</End>
        </Reservation>
      </PoolReservationLink>
    </SubscriptionIpRange>
    <SubscriptionIpRange>
      <SubscriptionIpRangeId>f186f259-cbe9-4bb9-b806-991795438c07</
SubscriptionIpRangeId>
      <FromAddress>abcd:1234:0:0:0:0:0:1</FromAddress>
      <ToAddress>abcd:1234:0:0:0:0:0:2</ToAddress>
    </SubscriptionIpRange>
    <SubscriptionIpRange>
      <SubscriptionIpRangeId>d9f38e4c-8589-413f-84ab-f5b721150353</
SubscriptionIpRangeId>
      <FromAddress>127.1.1.15</FromAddress>
      <ToAddress>127.1.1.16</ToAddress>
    </SubscriptionIpRange>
    <SubscriptionIpRange>
      <SubscriptionIpRangeId>9d11ff6e-f26e-4713-8355-e6cf786a5612</
SubscriptionIpRangeId>
      <FromAddress>127.1.1.17</FromAddress>
      <ToAddress>127.1.1.19</ToAddress>
    </SubscriptionIpRange>
  </IPRanges>
</SubscriptionIpRangesUpdated>
```

### Related reference

removePoolReservationLink() on page 213
addIpRange() on page 217
removeIpRange() on page 219

## SubscriptionPlanChanged

Smile emits this message when a new plan schedule item on the subscription's accounting plan schedule becomes active, either because its start date and time has elapsed or due to a modification to the plan schedule.

### Message header

The following table lists the parameters in the SubscriptionPlanChanged message header.

**Table 46: SubscriptionPlanChanged header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionPlanChanged" |

The message body contains the `<SubscriptionPlanChanged>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionPlanChanged
```

### Example: The `<SubscriptionPlanChanged>` element

This example shows a SubscriptionPlanChanged XML document that contains the `<SubscriptionPlanChanged>` element.

```
<SubscriptionPlanChanged xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
  <USN>2142426762</USN>
  <SID>170</SID>
  <ServiceName>Test service 170</ServiceName>
  <Parent>1000000008</Parent>
  <Properties>
    <Object>
      <String name="username">joe@example.com</String>
      <Boolean name="billPrintName">false</Boolean>
      <Boolean name="streetPrintName">true</Boolean>
    </Object>
  </Properties>
  <ProvisionStatus>Provisioned</ProvisionStatus>
  <SuspendStatus>Active</SuspendStatus>
  <ActivateStatus>Activated</ActivateStatus>
  <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <PlanScheduleItem>
    <PlanCode>tp170</PlanCode>
    <PlanScheduleType>1</PlanScheduleType>
    <Start>2012-08-08T00:00:00.000+10:00</Start>
  </PlanScheduleItem>
</SubscriptionPlanChanged>
```

#### Related reference

Plan.changePlan() on page 256

## `SubscriptionProvisionCommand`

If it is configured to do so, Smile emits this message to invoke a command in another application that provisions a new feature for an existing subscription.

The values for the command and commandID parameters are specific to each service and Smile implementation. Contact Inomial for more information about these parameters.

### Message header

The following table lists the parameters in the SubscriptionProvisionCommand message header.

**Table 47: SubscriptionProvisionCommand header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| command | string | The provisioning command to invoke |

| Parameter | Type | Values |
|---|---|---|
| commandID | integer | A parameter to the command |
| smileMessageType | string | "SubscriptionProvisionCommand" |

### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned, enabled and activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

## Related concepts

Message sequence—Provisioning on page 74

### SubscriptionProvisioned

Smile emits this message after the process to provision a subscription is complete.

A subscription must be provisioned, enabled and activated before it is fully available. These three events can happen in any order.

#### Message header

The following table lists the parameters in the SubscriptionProvisioned message header.

**Table 48: SubscriptionProvisioned header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| provisionRequestId | integer | The provision request ID number |
| smileMessageType | string | "SubscriptionProvisioned" |

#### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

#### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned and enabled, but is not yet activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
```

```
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

### Related concepts

Message sequence—Subscription orders on page 73

Message sequence—Provisioning on page 74

## SubscriptionProvisionResponse

An authoritative source emits this internal message to affect the provisioning status of a subscription. The authoritative source is a Smile application, but it is not part of Smile itself.

If Smile emits the `SubscriptionReadyToProvision` message, it must receive at least one `SubscriptionProvisionResponse` message before it can provision the subscription.

### Message header

The following table lists the parameters in the SubscriptionProvisionResponse message header.

**Table 49: SubscriptionProvisionResponse header parameters**

| Parameter | Type | Values |
|---|---|---|
| provisionRequestId | integer | The provision request ID number |
| smileMessageType | string | "SubscriptionProvisionResponse" |

### Message body

The message body contains the `<SubscriptionProvisionResponse>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionProvisionResponse
```

### Example: The `<SubscriptionProvisionResponse>` element

This example shows a SubscriptionProvisionResponse XML document that contains the `<SubscriptionProvisionResponse>` element.

```
<SubscriptionProvisionResponse xmlns="http://xml.inomial.com/
smile/2.xsd">
  <provisionRequestId>16876</provisionRequestId>
  <status>Provisioned</status>
  <message>Completed</message>
</SubscriptionProvisionResponse>
```

### Related concepts

Message sequence—Provisioning on page 74

### `SubscriptionReadyToProvision`

If configured to do so, Smile emits this message when a new subscription is ready to provision. This message may trigger network devices and upstream providers to begin their provisioning process. Smile does not emit this message by default.

A subscription must be provisioned, enabled and activated before it is fully available. These three events can happen in any order.

#### Message header

The following table lists the parameters in the SubscriptionReadyToProvision message header.

**Table 50: SubscriptionReadyToProvision header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| provisionRequestId | integer | The provision request ID number |
| smileMessageType | string | "SubscriptionReadyToProvision" |

#### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

#### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is activated but is not provisioned or enabled.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>ReadyToProvision</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>NotEnabled</EnableStatus>
```

```
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

Related concepts

Message sequence—Provisioning on page 74

## SubscriptionResumed

Smile emits this message after a subscription is unsuspended, re-activating billing and the subscription's access to a service.

### Message header

The following table lists the parameters in the SubscriptionResumed message header.

**Table 51: SubscriptionResumed header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionResumed" |

### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is provisioned, enabled and activated.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
```

```
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>Activated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

## `SubscriptionSpendLimitExceeded`

Smile emits this message when a subscription exceeds a spend limit. Listen for this message if you need to perform an action when a subscription exceeds a spend limit, for example, suspending access to international calls.

**Note:** Smile also emits the `SubscriptionSpendLimitStatusUpdated` message when a subscription exceeds a spend limit.

### Message header

The following table lists the parameters in the SubscriptionSpendLimitExceeded message header.

**Table 52: SubscriptionSpendLimitExceeded header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| spendLimitId | integer | The spend limit ID |
| currentThreshold | integer | The threshold limit that the subscription will next exceed |
| previousThreshold | integer | The threshold limit that the subscription has most recently exceeded |
| smileMessageType | string | "SubscriptionSpendLimitExceeded" |

### Message body

The message body contains the `<SubscriptionSpendLimits>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionSpendLimits
```

### Example: The `<SubscriptionSpendLimits>` element

This example shows a SubscriptionSpendLimits XML document that contains the `<SubscriptionSpendLimits>` element.

```
<SubscriptionSpendLimits xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
    <USN>2142425376</USN>
    <SID>137</SID>
    <ServiceName>Example Telecom Voice</ServiceName>
    <Parent>2142422670</Parent>
    <Properties>
      <Object>
        <String name="username">0491570156</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
        <Timestamp name="dob"/>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <AccessoryDataList/>
  </Subscription>
  <Limits>
    <SubscriptionSpendLimit>
      <SpendLimit>
        <spendLimitId>1</spendLimitId>
        <sid>137</sid>
        <name>Data</name>
        <limit>10</limit>
        <alertThresholds>
          <alertThreshold>50</alertThreshold>
          <alertThreshold>80</alertThreshold>
          <alertThreshold>100</alertThreshold>
        </alertThresholds>
      </SpendLimit>
      <periodEnd>2012-08-31+10:00</periodEnd>
      <effectiveLimit>10</effectiveLimit>
      <effectiveAlertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </effectiveAlertThresholds>
      <currentSpend>33.473000</currentSpend>
      <currentThreshold>100</currentThreshold>
      <previousThreshold>0</previousThreshold>
    </SubscriptionSpendLimit>
  </Limits>
</SubscriptionSpendLimits>
```

### Related concepts

Listen for spend limit changes on page 280

### SubscriptionSpendLimitReceded

Smile emits this message when a subscription that had exceeded a spend limit is within the spend limit again, for example, if the spend limit is reset to $0 at the start of the next rating period. Listen for this message if you need to perform an action when a subscription is again within a spend limit, for example, restoring access to international calls.

**Note:** Smile also emits the `SubscriptionSpendLimitStatusUpdated` message when a subscription is again within a spend limit.

#### Message header

The following table lists the parameters in the SubscriptionSpendLimitReceded message header.

**Table 53: SubscriptionSpendLimitReceded header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| spendLimitId | integer | The spend limit ID |
| currentThreshold | integer | The threshold limit that the subscription will next exceed |
| previousThreshold | integer | The threshold limit that the subscription has most recently exceeded |
| smileMessageType | string | "SubscriptionSpendLimitReceded" |

#### Message body

The message body contains the `<SubscriptionSpendLimits>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionSpendLimits
```

#### Example: The `<SubscriptionSpendLimits>` element

This example shows a SubscriptionSpendLimits XML document that contains the `<SubscriptionSpendLimits>` element.

```
<SubscriptionSpendLimits xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
    <USN>2142425376</USN>
    <SID>137</SID>
    <ServiceName>Example Telecom Voice</ServiceName>
    <Parent>2142422670</Parent>
    <Properties>
      <Object>
        <String name="username">0491570156</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
        <Timestamp name="dob"/>
      </Object>
    </Properties>
```

```
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <AccessoryDataList/>
  </Subscription>
  <Limits>
    <SubscriptionSpendLimit>
      <SpendLimit>
        <spendLimitId>1</spendLimitId>
        <sid>137</sid>
        <name>Data</name>
        <limit>10</limit>
        <alertThresholds>
          <alertThreshold>50</alertThreshold>
          <alertThreshold>80</alertThreshold>
          <alertThreshold>100</alertThreshold>
        </alertThresholds>
      </SpendLimit>
      <periodEnd>2012-08-31+10:00</periodEnd>
      <effectiveLimit>10</effectiveLimit>
      <effectiveAlertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </effectiveAlertThresholds>
      <currentSpend>33.473000</currentSpend>
      <currentThreshold>100</currentThreshold>
      <previousThreshold>0</previousThreshold>
    </SubscriptionSpendLimit>
  </Limits>
</SubscriptionSpendLimits>
```

### Related concepts

Listen for spend limit changes on page 280

## SubscriptionSpendLimitStatusUpdated

Smile emits this message after a subscription crosses a spend limit alert threshold, or when a new rating period begins and all spend limits for the subscription are reset. For example, alert thresholds may be configured when a subscription consumes 60%, 80% and 100% of a spend limit.

Listen for this message if you need to send an alert to a customer when a subscription crosses an alert threshold. To avoid sending alerts when the spend limit is reset, filter the messages using the `currentThreshold` and `previousThreshold` parameters.

**Note:** Smile may also emit the `SubscriptionSpendLimitExceeded` or `SubscriptionSpendLimitReceded` message at the same time it emits this message.

### Message header

The following table lists the parameters in the SubscriptionSpendLimitStatusUpdated message header.

**Table 54: SubscriptionSpendLimitStatusUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| spendLimitId | integer | The spend limit ID |
| currentThreshold | integer | The threshold limit that the subscription will next exceed |
| previousThreshold | integer | The threshold limit that the subscription has most recently exceeded |
| smileMessageType | string | "SubscriptionSpendLimitStatusUpdated" |

### Message body

The message body contains the `<SubscriptionSpendLimits>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionSpendLimits
```

### Example: The `<SubscriptionSpendLimits>` element

This example shows a SubscriptionSpendLimits XML document that contains the `<SubscriptionSpendLimits>` element.

```
<SubscriptionSpendLimits xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
    <USN>2142425376</USN>
    <SID>137</SID>
    <ServiceName>Example Telecom Voice</ServiceName>
    <Parent>2142422670</Parent>
    <Properties>
      <Object>
        <String name="username">0491570156</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
        <Timestamp name="dob"/>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
    <AccessoryDataList/>
  </Subscription>
  <Limits>
    <SubscriptionSpendLimit>
      <SpendLimit>
        <spendLimitId>1</spendLimitId>
        <sid>137</sid>
        <name>Data</name>
        <limit>10</limit>
        <alertThresholds>
          <alertThreshold>50</alertThreshold>
```

```
          <alertThreshold>80</alertThreshold>
          <alertThreshold>100</alertThreshold>
        </alertThresholds>
      </SpendLimit>
      <periodEnd>2012-08-31+10:00</periodEnd>
      <effectiveLimit>10</effectiveLimit>
      <effectiveAlertThresholds>
        <alertThreshold>50</alertThreshold>
        <alertThreshold>80</alertThreshold>
        <alertThreshold>100</alertThreshold>
      </effectiveAlertThresholds>
      <currentSpend>33.473000</currentSpend>
      <currentThreshold>100</currentThreshold>
      <previousThreshold>0</previousThreshold>
    </SubscriptionSpendLimit>
  </Limits>
</SubscriptionSpendLimits>
```

Related concepts

Listen for spend limit changes on page 280

Related reference

setSpendLimitOverride() on page 275

setTemporarySpendLimitOverride() on page 276

setSpendLimitAlertThresholdOverrides() on page 278

## SubscriptionSuspended

Smile emits this message after a subscription's access to a service is temporarily suspended.

### Message header

The following table lists the parameters in the SubscriptionSuspended message header.

**Table 55: SubscriptionSuspended header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionSuspended" |

### Message body

The message body contains the `<SubscriptionFeaturesUpdate>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionFeaturesUpdate
```

### Example: The `<SubscriptionFeaturesUpdate>` element

This example shows a SubscriptionFeaturesUpdate XML document that contains the `<SubscriptionFeaturesUpdate>` element. Note that the subscription is suspended.

```
<SubscriptionFeaturesUpdate xmlns="http://xml.inomial.com/
smile/2.xsd">
  <Subscription>
    <USN>2142426762</USN>
    <SID>170</SID>
    <ServiceName>Test service 170</ServiceName>
    <Parent>1000000008</Parent>
    <Properties>
      <Object>
        <String name="username">joe@example.com</String>
        <Boolean name="billPrintName">false</Boolean>
        <Boolean name="streetPrintName">true</Boolean>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Suspended</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
  <ActiveFeatures/>
  <PlanScheduleItems>
    <PlanScheduleItem>
      <PlanCode>tp170</PlanCode>
      <PlanScheduleType>1</PlanScheduleType>
      <Start>2012-08-08T00:00:00.000+10:00</Start>
    </PlanScheduleItem>
  </PlanScheduleItems>
</SubscriptionFeaturesUpdate>
```

#### Related reference

suspend() on page 177
resume() on page 179

### SubscriptionUpdated

Smile emits this message after subscription data is changed.

This message contains the current data for a subscription, such as its properties and status, but it does not specify what has been changed or removed. To determine which properties have changed, listeners must persist their own set of properties and compare this set to the set in the message.

#### Message header

The following table lists the parameters in the SubscriptionUpdated message header.

**Table 56: SubscriptionUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| sid | integer | The service ID |
| usn | string | The unique subscription ID number for the subscription |
| smileMessageType | string | "SubscriptionUpdated" |

#### Message body

The message body contains the `<SubscriptionUpdated>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}SubscriptionUpdated
```

#### Example: The `<SubscriptionUpdated>` element

This example shows a SubscriptionUpdated XML document that contains the `<SubscriptionUpdated>` element.

```
<SubscriptionUpdated xmlns="http://xml.inomial.com/smile/2.xsd">
  <Subscription>
    <USN>2142426697</USN>
    <SID>17</SID>
    <ServiceName>Smile Access</ServiceName>
    <Parent>2142426689</Parent>
    <Properties>
      <Object>
        <String name="emailAddress">joe@example.com</String>
        <String name="username">joe@example.com</String>
      </Object>
    </Properties>
    <ProvisionStatus>Provisioned</ProvisionStatus>
    <SuspendStatus>Active</SuspendStatus>
    <ActivateStatus>NotActivated</ActivateStatus>
    <EnableStatus>Enabled</EnableStatus>
  </Subscription>
</SubscriptionUpdated>
```

Related reference

update() on page 171

changeUsername() on page 174

## SmileTransaction

Smile emits messages in the SmileTransaction to notify listeners of transaction changes.

The messages in SmileTransaction are:

- `TransactionAllocated`
- `TransactionClosed`
- `TransactionDeleted`
- `TransactionReopened`
- `TransactionReversed`
- `TransactionUpdated`

Related concepts

Transaction lifecycle on page 379
Synchronise transaction information on page 380

### `TransactionAllocated`

Smile emits this message when a transaction is allocated to another transaction, for example, a receipt is allocated to an invoice. A transaction can have more than one allocated transaction against it.

#### Message header

The following table lists the parameters in the TransactionAllocated message header.

**Table 57: TransactionAllocated header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |
| debitNumber | string | The transaction number of the debit transaction in this allocation event |
| creditNumber | string | The transaction number of the credit transaction in this allocation event |
| allocationType | string | "Allocation" or "Deallocation" |
| smileMessageType | string | "TransactionAllocated" |
| eventType | string | "TransactionAllocated" |

### Message body

The message body contains the `TransactionAllocatedEvent` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionAllocatedEvent
```

### Example: The `<TransactionAllocatedEvent>` element

This example shows a TransactionAllocatedEvent XML document that contains the `<TransactionAllocatedEvent>` element.

```
<TransactionAllocatedEvent xmlns="http://xml.inomial.com/
smile/2.xsd">
  <transactionEventType>TransactionAllocated</transactionEventType>
  <debitTransaction>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>21435540</transactionNumber>
    <currency>AUD</currency>
    <amount>221.55</amount>
    <accountType>1</accountType>
    <gstAmount>20.14</gstAmount>
    <entryTimestamp>2012-07-31T10:00:00.000+10:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$221.55">221.55</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$20.14">20.14</
formattedGstAmount>
    <unallocatedAmount>221.55</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2012-08-09T13:52:44.629+10:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$221.55">221.55</formattedUnallocatedAmount>
  </debitTransaction>
  <creditTransaction>
    <version>2.0</version>
    <transactionType>Receipt</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>RCPT1000265</transactionNumber>
    <currency>AUD</currency>
    <amount>-221.55</amount>
    <accountType>1</accountType>
    <gstAmount>0.00</gstAmount>
    <entryTimestamp>2012-08-09T14:17:19.683+10:00</entryTimestamp>
    <salesReportCategory>receipt</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$(221.55)">-221.55</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
    <unallocatedAmount>-221.55</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2012-08-09T14:17:19.685+10:00</logTimestamp>
```

```
    <formattedUnallocatedAmount currency="AUD"
 formatted="$(221.55)">-221.55</formattedUnallocatedAmount>
  </creditTransaction>
  <amount>221.55</amount>
</TransactionAllocatedEvent>
```

### Related concepts

Transaction overview on page 374
Transaction types on page 376

### Related reference

allocate() on page 108
releaseBond() on page 154
processPayment() on page 291

## TransactionClosed

Smile emits this message when a transaction is closed and the transaction value is added to the account balance.

Transactions may be open for some time before they are closed, for example, quotes or credit card payments pending bank approval. Other transactions are closed immediately, for example, invoices, receipts for cash payments and adjustments.

### Message header

The following table lists the parameters in the TransactionClosed message header.

**Table 58: TransactionClosed header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |
| transactionNumber | string | The transaction number that Smile generates to identify the transaction |
| transactionType | string | "CreditNote", "CreditTransferCredit", "CreditTransferDebit", "DebitNote", "Invoice", "InvoiceReversal", "ItemisedCredit", "ItemisedCreditReversal", "ItemisedDebit", "ItemisedDebitReversal", "Quote", "Receipt", "ReceiptReversal", "RecipientCreatedTaxInvoice", "RecipientCreatedTaxInvoiceReversal", "Surcharge", "SurchageReversal", "TransferredCredit" or "TransferredDebit" |

| Parameter | Type | Values |
|---|---|---|
| smileMessageType | string | "TransactionClosed" |
| eventType | string | "TransactionClosed" |

### Message body

For itemised transactions (invoices, RCTIs and itemised credits and debits), the message body contains the `<ItemisedTransactionDetailEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}ItemisedTransactionDetailEvent
```

For all other transactions, the message body contains the `<TransactionDetailEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionDetailEvent
```

### Example: The `<TransactionDetailEvent>` element

This example shows a TransactionDetailEvent XML document that contains the `<TransactionDetailEvent>` element. This example is for a non-itemised (receipt) transaction.

```
<TransactionDetailEvent xmlns="http://xml.inomial.com/smile/2.xsd">
  <transactionEventType>TransactionClosed</transactionEventType>
  <transaction>
    <version>2.0</version>
    <transactionType>Receipt</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>RCPT1000265</transactionNumber>
    <currency>AUD</currency>
    <amount>-221.55</amount>
    <accountType>1</accountType>
    <gstAmount>0</gstAmount>
    <entryTimestamp>2012-08-09T14:17:19.683+10:00</entryTimestamp>
    <salesReportCategory>receipt</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$(221.55)">-221.55</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0</
formattedGstAmount>
    <unallocatedAmount>-221.55</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2012-08-09T14:17:19.685+10:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$(221.55)">-221.55</formattedUnallocatedAmount>
  </transaction>
</TransactionDetailEvent>
```

## Related concepts

Transaction types on page 376

## Related reference

raiseInvoice() on page 102

raiseAdjustment() on page 113

raiseReimbursement() on page 121
reverseTransaction() on page 123
createBond() on page 152
releaseBond() on page 154
Plan.changePlan() on page 256
processPayment() on page 291
commitContractTermination() on page 346

### `TransactionDeleted`

Smile emits this message when it deletes a transaction, removing the transaction value from the account balance and the transaction from the account history. However, Smile records the delete action in the logs to maintain an audit trail of the event.

You cannot delete a transaction if the deletion affects the integrity of other Smile data.

#### Message header

The following table lists the parameters in the TransactionDeleted message header.

**Table 59: TransactionDeleted header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |
| transactionNumber | string | The transaction number that Smile generates to identify the transaction |
| transactionType | string | "CreditNote", "CreditTransferCredit", "CreditTransferDebit", "DebitNote", "Invoice", "InvoiceReversal", "ItemisedCredit", "ItemisedCreditReversal", "ItemisedDebit", "ItemisedDebitReversal", "Receipt", "ReceiptReversal", "RecipientCreatedTaxInvoice", "RecipientCreatedTaxInvoiceReversal", "Surcharge", "SurchargeReversal", "TransferredCredit" or "TransferredDebit" |
| smileMessageType | string | "TransactionDeleted" |
| eventType | string | "TransactionDeleted" |

#### Message body

The message body contains the `<TransactionEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionEvent
```

### Example: The `<TransactionEvent>` element

This example shows a TransactionEvent XML document that contains the `<TransactionEvent>` element.

```
<TransactionEvent xmlns="http://xml.inomial.com/smile/2.xsd">
  <transactionEventType>TransactionDeleted</transactionEventType>
  <transaction>
    <version>2.0</version>
    <transactionType>Receipt</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>RCPT1000261</transactionNumber>
    <currency>AUD</currency>
    <amount>-108.90</amount>
    <accountType>1</accountType>
    <gstAmount>0.00</gstAmount>
    <entryTimestamp>2012-08-09T12:38:58.383+10:00</entryTimestamp>
    <salesReportCategory>receipt</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$(108.90)">108.90</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$0.00">0.00</
formattedGstAmount>
  </transaction>
</TransactionEvent>
```

#### Related concepts

Transaction types on page 376

## `TransactionReopened`

Smile emits this message when it reopens a transaction and any items in the transaction, removing the transaction value from the account balance. You can now make financial changes to the transaction.

### Message header

The following table lists the parameters in the TransactionReopened message header.

**Table 60: TransactionReopened header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |

| Parameter | Type | Values |
|---|---|---|
| transactionNumber | string | The transaction number that Smile generates to identify the transaction |
| transactionType | string | "CreditNote", "CreditTransferCredit", "CreditTransferDebit", "DebitNote", "Invoice", "InvoiceReversal", "ItemisedCredit", "ItemisedCreditReversal", "ItemisedDebit", "ItemisedDebitReversal", "Receipt", "ReceiptReversal", "RecipientCreatedTaxInvoice", "RecipientCreatedTaxInvoiceReversal", "Surcharge", "SurchargeReversal", "TransferredCredit" or "TransferredDebit" |
| smileMessageType | string | "TransactionReopened" |
| eventType | string | "TransactionReopened" |

### Message body

The message body contains the `<TransactionEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionEvent
```

### Example: The `<TransactionEvent>` element

This example shows a TransactionEvent XML document that contains the `<TransactionEvent>` element.

```
<TransactionEvent xmlns="http://xml.inomial.com/smile/2.xsd">
  <transactionEventType>TransactionReopened</transactionEventType>
  <transaction>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>21438650</transactionNumber>
    <currency>AUD</currency>
    <amount>108.90</amount>
    <accountType>1</accountType>
    <gstAmount>9.90</gstAmount>
    <entryTimestamp>2012-08-09T12:38:07.937+10:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$108.90">108.90</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$9.90">9.90</
formattedGstAmount>
  </transaction>
</TransactionEvent>
```

## Related concepts

Transaction types on page 376

### `TransactionReversed`

Smile emits this message when it reverses a transaction, that is, raises a transaction that negates another transaction.

#### Message header

The following table lists the parameters in the TransactionReversed message header.

**Table 61: TransactionReversed header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |
| transactionNumber | string | The transaction number that Smile generates to identify the transaction |
| transactionType | string | "CreditNote", "CreditTransferCredit", "CreditTransferDebit", "DebitNote", "Invoice", "InvoiceReversal", "ItemisedCredit", "ItemisedCreditReversal", "ItemisedDebit", "ItemisedDebitReversal", "Receipt", "ReceiptReversal", "RecipientCreatedTaxInvoice", "RecipientCreatedTaxInvoiceReversal", "Surcharge", "SurchargeReversal", "TransferredCredit" or "TransferredDebit" |
| smileMessageType | string | "TransactionReversed" |
| eventType | string | "TransactionReversed" |

#### Message body

The message body contains the `<TransactionEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionEvent
```

#### Example: The `<TransactionEvent>` element

This example shows a TransactionEvent XML document that contains the `<TransactionEvent>` element.

```
<TransactionEvent xmlns="http://xml.inomial.com/smile/2.xsd">
  <transactionEventType>TransactionReversed</transactionEventType>
  <transaction>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
```

```
    <transactionNumber>21435565</transactionNumber>
    <currency>AUD</currency>
    <amount>218.30</amount>
    <accountType>1</accountType>
    <gstAmount>19.85</gstAmount>
    <entryTimestamp>2012-07-31T10:00:00.000+10:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$218.30">218.30</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$19.85">19.85</
formattedGstAmount>
  </transaction>
  <effectiveDate>2012-08-09+10:00</effectiveDate>
</TransactionEvent>
```

### Related concepts

Transaction types on page 376

Reversals on page 378

### Related reference

reverseTransaction() on page 123

## TransactionUpdated

Smile emits this message in two circumstances:

- A transaction is not reopened, but the due date or comments are changed. Smile does not change the transaction value.
- A reopened transaction is closed. Smile adds the new transaction value to the account balance.

### Message header

The following table lists the parameters in the TransactionUpdated message header.

**Table 62: TransactionUpdated header parameters**

| Parameter | Type | Values |
|---|---|---|
| version | string | "2.0" |
| usn | string | The unique subscription number for the account that the transaction applies to |
| company | integer | The company ID this account belongs to |
| currency | string | The ISO 4217 currency code of this account and transaction |
| amount | decimal | The value of this transaction |
| accountType | integer | The ID of the account type, as defined in Smile |
| transactionNumber | string | The transaction number that Smile generates to identify the transaction |

| Parameter | Type | Values |
|---|---|---|
| transactionType | string | "CreditNote", "CreditTransferCredit", "CreditTransferDebit", "DebitNote", "Invoice", "InvoiceReversal", "ItemisedCredit", "ItemisedCreditReversal", "ItemisedDebit", "ItemisedDebitReversal", "Receipt", "ReceiptReversal", "RecipientCreatedTaxInvoice", "RecipientCreatedTaxInvoiceReversal", "Surcharge", "SurchargeReversal", "TransferredCredit" or "TransferredDebit" |
| smileMessageType | string | "TransactionUpdated" |
| eventType | string | "TransactionUpdated" |

### Message body

For itemised transactions (invoices, RCTIs and itemised credits and debits), the message body contains the `<ItemisedTransactionDetailEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}ItemisedTransactionDetailEvent
```

For all other transactions, the message body contains the `<TransactionDetailEvent>` element and has the following format:

```
{http://xml.inomial.com/smile/2.xsd}TransactionDetailEvent
```

### Example: The `<ItemisedTransactionDetailEvent>` element

This example shows an ItemisedTransactionDetailEvent XML document that contains the `<ItemisedTransactionDetailEvent>` element. This example is for an itemised (invoice) transaction.

```
<ItemisedTransactionDetailEvent xmlns="http://xml.inomial.com/
smile/2.xsd">
  <transactionEventType>TransactionUpdated</transactionEventType>
  <transaction>
    <version>2.0</version>
    <transactionType>Invoice</transactionType>
    <company>1</company>
    <usn>1000000008</usn>
    <transactionNumber>21435540</transactionNumber>
    <currency>AUD</currency>
    <amount>221.55</amount>
    <accountType>1</accountType>
    <gstAmount>20.14</gstAmount>
    <entryTimestamp>2012-07-31T10:00:00.000+10:00</entryTimestamp>
    <salesReportCategory>invoice</salesReportCategory>
    <formattedAmount currency="AUD" formatted="$221.55">221.55</
formattedAmount>
    <formattedGstAmount currency="AUD" formatted="$20.14">20.14</
formattedGstAmount>
    <unallocatedAmount>0.00</unallocatedAmount>
    <enteredByUsn>1000000016</enteredByUsn>
    <logTimestamp>2012-08-09T13:52:44.629+10:00</logTimestamp>
    <formattedUnallocatedAmount currency="AUD"
 formatted="$0.00">0.00</formattedUnallocatedAmount>
    <transactionDate>2012-07-31+10:00</transactionDate>
```

```
    <dueDate>2012-08-21+10:00</dueDate>
    <openDate>2012-07-31+10:00</openDate>
    <closeDate>2012-08-09+10:00</closeDate>
    <quoteNumber>21435540</quoteNumber>
    <createdByUsn>1000000016</createdByUsn>
    <discount>0.00</discount>
    <transactionSource>Bulk</transactionSource>
    <transactionItem>
      <usn>2142426499</usn>
      <lineNumber>0</lineNumber>
      <itemCode>007459</itemCode>
      <chargeFrom>2012-07-17+10:00</chargeFrom>
      <chargeTo>2012-07-30+10:00</chargeTo>
      <description>Invoice item description</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>2.26</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.230000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142426499</usn>
      <lineNumber>1</lineNumber>
      <itemCode>007460</itemCode>
      <chargeFrom>2012-07-17+10:00</chargeFrom>
      <chargeTo>2012-07-30+10:00</chargeTo>
      <description>Invoice item description</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>4.52</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.450000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142426499</usn>
      <lineNumber>2</lineNumber>
      <itemCode>007458</itemCode>
      <chargeFrom>2012-07-31+10:00</chargeFrom>
      <chargeTo>2012-07-31+10:00</chargeTo>
      <description>Early termination charge</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>194.62</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>19.460000</gstAmount>
      <salesReportCategory>unknown</salesReportCategory>
```

```
    </transactionItem>
    <transactionItem>
      <usn>2142426499</usn>
      <lineNumber>3</lineNumber>
      <itemCode>round</itemCode>
      <chargeFrom>2012-07-31+10:00</chargeFrom>
      <chargeTo>2012-07-31+10:00</chargeTo>
      <description>Rounding adjustment</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>0.01</amount>
      <isGst>false</isGst>
      <chargeGst>true</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>round</salesReportCategory>
    </transactionItem>
    <transactionItem>
      <usn>2142426499</usn>
      <lineNumber>4</lineNumber>
      <itemCode>gst</itemCode>
      <chargeFrom>2012-07-31+10:00</chargeFrom>
      <chargeTo>2012-07-31+10:00</chargeTo>
      <description>Includes 10% GST</description>
      <quantity>1.000</quantity>
      <currency>AUD</currency>
      <amount>20.14</amount>
      <isGst>true</isGst>
      <chargeGst>false</chargeGst>
      <discount>0.00</discount>
      <count>0</count>
      <gstAmount>0.000000</gstAmount>
      <salesReportCategory>gst</salesReportCategory>
    </transactionItem>
  </transaction>
</ItemisedTransactionDetailEvent>
```


## Related concepts

Transaction types on page 376

# Appendix C

## Faults common to all SOAP methods

An exception message is returned when an API call fails for a particular reason. Specific exceptions are documented against each SOAP method.

The following exceptions apply to all documented SOAP methods.

### AuthenticationException

This fault is returned when Smile cannot authenticate the identity of the API caller.

### AccessException

This fault is returned when the API caller does not have permission to perform the requested function.

# Appendix D

## Smile architecture

The following diagram provides an overview of Smile's internal components and external interfaces.

**Figure 14: Smile architecture**

# Appendix E

## Support

If you have trouble interacting with Smile, email support@inomial.com for assistance. Provide as much information as you can about the problem in your email, including:

- details of the account or user affected
- examples of other users affected
- any system-generated output about the error, such as error messages or log reports
- when the problem occurred
- level of urgency
- your contact details

### Emergency support

For emergency support, call Inomial's 24-hour support line on +61 3 9663 3554. Emergency support is only available if the problem affects more than one end user. Please call, not email, to advise us of urgent problems so we can quickly resolve the issue.

If your call is after hours, listen to the recorded instructions on the support line, call the number provided and leave a number on which we can contact you personally. If you have an outage, ensure that the number you leave can accept calls.

# Glossary

**account**

A customer, company or debtor in the accounts receivable system to whom one or more subscriptions are provided. For example, you can provide mobile subscription 0401234567 to the account of Joe Smith. The account defines who bills are sent to, how much the customer owes and their payment terms.

**account terms**

The credit control settings for an account. Account terms specify the duration of treatment levels and the action taken at each treatment level, among other settings.

**account type**

A reporting concept that helps you report on different groups of accounts in your company. For example, you may have different account types for different companies that you have acquired or for GST exempt and non-GST exempt accounts.

**activity statement**

An itemised list of subscription usage over a period of time.

**ACMA**

The Australian Communications and Media Authority. ACMA is an independent statutory authority tasked with ensuring most elements of Australia's media and communications legislation, related regulations, and numerous derived standards and codes of practice operate effectively and efficiently, and in the public interest.

**adjustment**

A transaction that increases or decreases the amount charged for another transaction.

**agent**

An entity that associates a commission structure with a commission earner.

**alert threshold**

A percentage of value that, when consumed, triggers a notification and raises a message. For example, you can configure alert thresholds at 60%, 80% and 100% of a value pool. Alert thresholds apply to spend limits, usage alert groups and value pools.

**BSS**

Business support system. A system that manages your customer-facing operations, for example, creating orders, tracking payments and managing plans.

**CAT**

Customer access toolkit. Smile's online customer portal that lets customers view their usage history, download invoices and pay online.

**CDR**

Call detail record. The usage data associated with a phone call, for example, destination, tariff type and duration.

**commission earner**

A person who is paid commission. Each commission earner is added to at least one agent and associated with a commission structure.

**commission payment delay**

The period of time between when an invoice is fully paid and the commission is raised. The commission payment delay ensures that you do not pay commission on invalid payments.

**commission structure**
The commission rates that you pay to a commission earner.

**credit control**
A series of actions that Smile automatically applies when an unpaid account reaches a treatment level.

**credit limit**
The account balance at which a customer's access to one or more subscriptions is suspended. When calculating a credit limit, Smile includes the value of outstanding account balances and any usage in the current billing cycle. The credit limit includes all subscriptions for an account. For example, if an account has a mobile subscription and a DSL subscription and the customer exceeds the credit limit on the account, Smile suspends both subscriptions.

**debtor close date**
A date that prevents changes to earlier transactions and ensures long-term report stability. You cannot change a transaction that was raised earlier than the debtor close date. The debtor close date lags behind real time to give you a window of time to correct mistakes before the transaction is locked.

**deprecated**
Items marked as deprecated are obsolete, no longer in use or being phased out.

**disposition**
The debtor status of an account. An account's disposition is also known as its treatment level.

**feature**
An item that modifies a subscription and that can be switched on and off, for example, voicemail, call forwarding or DSL shaping. Features have their own schedule, for example, the start and end dates for the period that call forwarding is active on a subscription.

**invoice**
A list of charges raised against a subscription or account over a given period of time. The invoice may contain usage charges, recurring charges, GST or other taxes, discounts and rounding amounts. Each invoice is associated with an account, although the invoice may contain charges for any subscriptions associated with the account.

**invoicing cycle**
Specifies when an invoice is generated. When you perform a billing run, Smile generates an invoice for all accounts that have subscriptions with an invoicing period that ends on the date of the billing run.

**invoicing period**
An instance of an invoicing cycle. For example, a subscription may have an invoicing period that ends on the 1st of each month and an invoicing period of 2 August-1 September.

**IPDR**
Internet protocol detail record. The usage data associated with an internet session, for example, duration, bytes uploaded and bytes downloaded.

**operator**
An account that can access Smile. For example, customer service representatives (CSRs), helpdesk staff and accounting staff may all be operators.

**OSS**
Operational support system. A system that manages the network services that support your business, for example, provisioning and customer access to the network.

**package**
A bundle of services added as a group to an account. Packages specify default values for bill layout and billing frequencies, and are used to order subscriptions and accounts.

**payment plan**
An agreed series of payments that lets customers pay outstanding invoices in installments. Smile suspends credit control actions against each invoice in a payment plan.

**PCI DSS**
The Payment Card Industry Data Security Standard (PCI DSS) is a proprietary information security standard for organizations that handle branded credit cards from the major card brands.

**plan**
The set of rules that determine how much is charged for a subscription. A plan can include recurring charges, usage charges, minimum charges, included usage and access rules such as modem speed.

**plan schedule**
Specifies the plans a subscription uses throughout its lifecycle. The plan schedule is made up of plan schedule items.

**plan schedule item (PSI)**
Specifies when a plan becomes active for a subscription.

**pre-rating**
A process that runs at intervals in open periods and provides an indication of subscription usage in the period to date.

**rating cycle**
Specifies the period for which usage charges are calculated. Rating periods can end at the end of each month or at a preset, recurring interval after the subscription was created.

**rating period**
An instance of a rating cycle. For example, a subscription may have a rating period that ends on the 1st of each month and a rating period of 2 August-1 September.

**RCTI**
Recipient created tax invoice. An invoice from a creditor requesting payment that is created by and addressed to a company. Smile generates RCTIs on behalf of creditors to manage commission payments. RCTIs include GST and other tax where applicable.

**recurring charge**
A regular charge that is not based on usage, for example, an insurance surcharge.

**service**
A generic kind of product that can be provided to customers, for example, ADSL or mobile telephony.

**setup charge**
An initial charge to add a customer to a plan, for example, an ADSL connection fee.

**SID**
Service ID. A number that uniquely identifies each Smile service.

**spend limit**
A limit that tracks the amount a subscription spends on a particular type of usage in each rating period, and lets you take action when a customer exceeds spend limit alert thresholds. For example, you can define a spend limit of $100 on premium SMS for a mobile service. Default spend limits are defined on a service, but you can override and customise the default spend limits for a subscription.

**stationery**
The format of printed documents. Smile's stationery functionality lets you specify the logos, background images and layout for invoices, journal entries, receipts and statements.

**subscription**
A single instance of a service that is provided to a customer, for example, a single mobile number. Subscriptions belong to an account and are often the source of usage and recurring charges.

**SZU**
Standard zone unit. An SZU is a geographic area of a charging zone specified by ACMA. Australia is divided into almost 2100 SZUs.

**treatment level**
The debtor status of an account. Smile has three default treatment levels: settled (account is paid before the due date), overdue (the account is overdue by 14 days or less) and delinquent (the account is overdue by 15 days or more), although you can configure additional treatment levels. An account's treatment level is also known as its disposition.

**UO**
User object. Related to custom fields, attributes and forms.

**usage alert group**
The physical amount of included usage in a plan. For example, you can define a usage alert group for 50 GB of included data on a ADSL plan, or 100 included SMS on a mobile plan.

**usage charge**
A charge that specifies how much to charge for different blocks of continuous use, for example, telephone flagfall and timed charges, the number of pages in a fax transmission or mobile internet downloads.

**USN**
Unique subscription number. A number that uniquely identifies each account and subscription in your database. A USN is associated with the same account or subscription throughout its lifecycle and is not reused after the account or subscription is closed.

**value pool**
The monetary amount of included usage in a plan.

**walled garden**
An environment that limits the content and services a customer can access.

**workflow**
The steps through which a process moves as it progresses from initiation to completion.

**xDR**
A generic term for a usage record for a service. For example, CDRs and IPDRs are both types of xDRs.