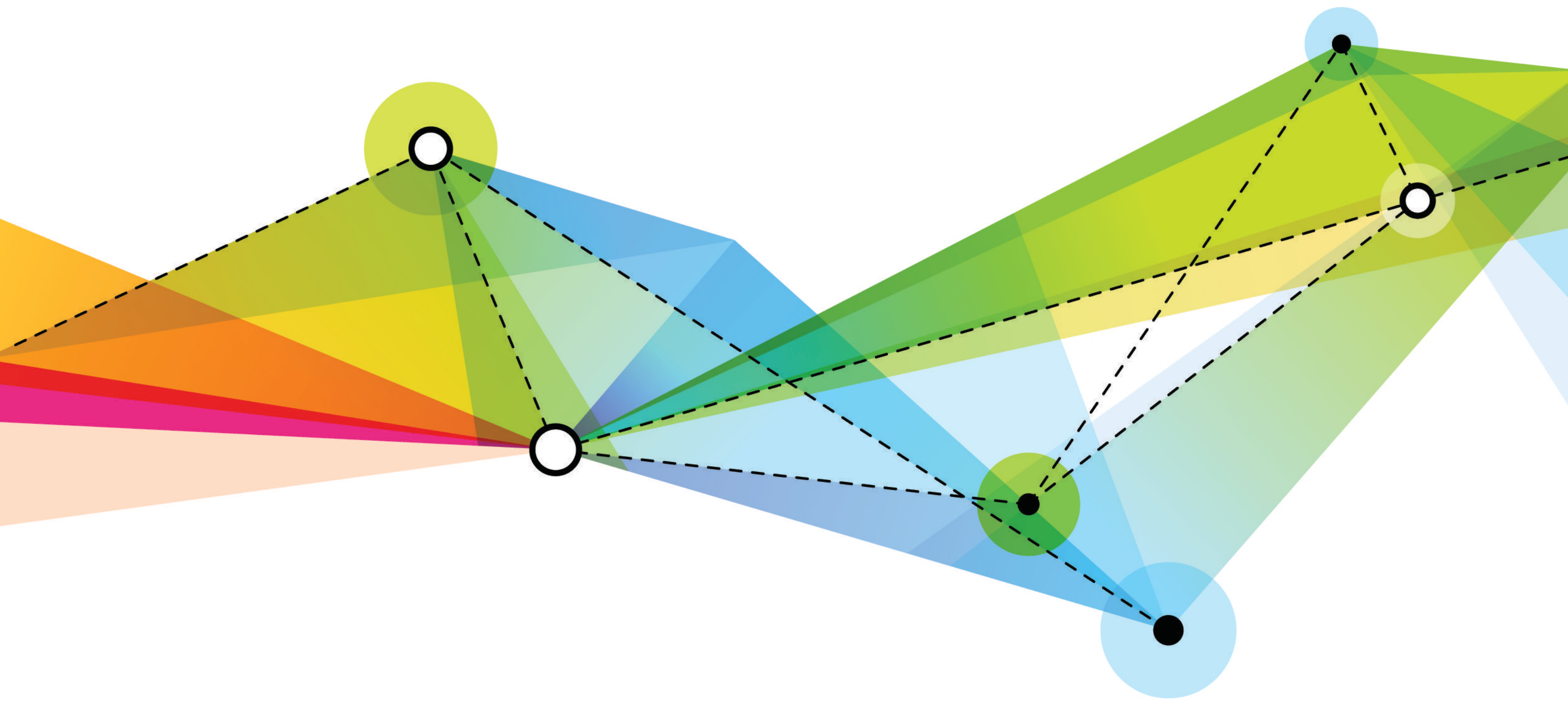
inomial

Flexible CSV CDR Importer



Edition: 1.0
Release date: August 17, 2016
Smile version: 6.0

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

Flexible CSV CDR Importer

Introduction

The flexible CSV CDR importer is a method of importing CSV CDR files into Smile without requiring a custom importer for the task. The flexible importer provides the ability to inform Smile what it should do with each column of a CSV - where in Smile the column is mapped, how to parse or convert the field information.

The flexible importer uses JExpr to express how Smile should handle an imported CSV column. For more information, see the *JExpr Language Guide*.

This document describes how to configure a Smile flexible importer to process a CSV CDR import file.

Note: JExpr is intended for advanced users familiar with programming languages. Contact Inomial for assistance.

Importers

Importers are used for importing usage data into Smile to complete rating and invoice generation.

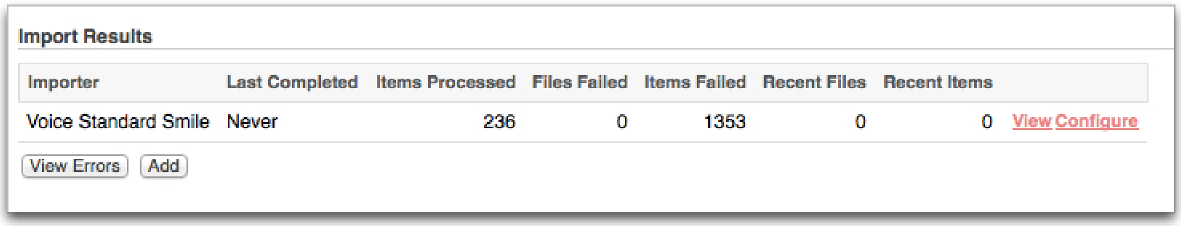
A configured importer specifies the importer type, location or method of retrieving usage data files, configuration options applied during import, error actions, relevant services and the task schedule of the import action.

To view Smile Importers, select **Importers** under **Services, Ordering & Rating** on the Configuration and Tools page.

Importer types are internally defined in Smile. Importer types are advanced configuration. For more information, contact Inomial.

The Smile Importers main page displays a summary list of configured importers and import results. The following actions can be taken on configured importers:

- **View**—displays current import results for an importer, including the number of successful and failed file and item events.
- **Configure**—displays the configuration of the importer. The configuration of an importer can be edited.
- **View Errors**—displays import errors by importer, failure type and missing subscriptions. For more information about how to check for failed import items, see the *User Guide*.


Import Results

Importer	Last Completed	Items Processed	Files Failed	Items Failed	Recent Files	Recent Items	
Voice Standard Smile	Never	236	0	1353	0	0	View Configure

Figure 1: Smile Importers

Add an importer

Smile importers define how Smile gets and handles usage data. An appropriate **Importer Type** is required to add an importer.

This task explains how to add an importer.

1. Select **Importers** under **Services, Ordering & Rating** on the Configuration and Tools page.

   The Smile Importers page is displayed.
2. Click **Add**.

   A **New Importer** window is displayed.

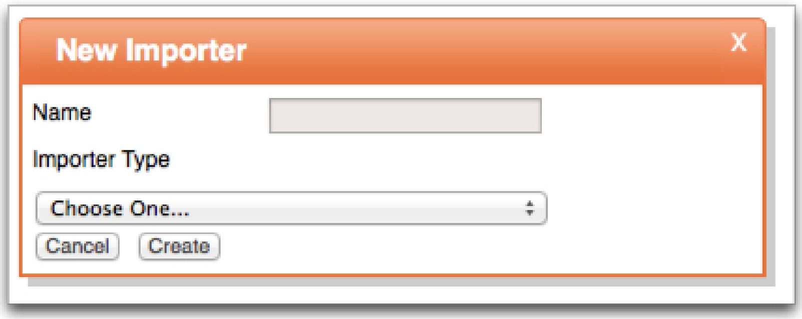


Figure 2: New Importer window

3. Type a name for the importer in the **Name** field.
4. Select **Flexible CSV CDR Importer** from the **Importer Type** drop-down.

   Importer Types are internally defined. For more information, contact Inomial.
5. Click **Create**.

   The Configure Importer page is displayed.
6. Complete the required importer configuration fields.
7. Click **Manage Scheduled Task** to define when and how often the importer runs.

   For more information, see the *Configuration Guide*.
8. Click **Save**.

   The Smile Importers page is displayed. The importer is added to the **Import Results** summary list.

Importer field properties

The following fields are available to configure non-JExpr properties of the flexible importer.

Name	Specifies a descriptive name of the importer.
Spool subdirectory for fetched files	Specifies the Smile subdirectory in which Smile places files to be processed.
Remote Hostname	Specifies the hostname of the remote server from which Smile retrieves the CDR CSV.
Remote port	Specifies the port number of the remote server from which Smile retrieves the CDR CSV. Default port number is 22.
Remote username	Specifies the username that Smile provides to the remote host.
Filename of local private key (id_dsa)	Specifies the name of the file containing the private key Smile users to authenticate the SFTP session.
Remote directory containing CDRs	Specifies the directory on the remote server from which Smile begins fetching files.
Remote filename regex	Specifies the regular expression defining which files to fetch. Importer will only download files matching this pattern. Leave blank to use the default.
Traverse the remote directory recursively	Specifies that the importer will traverse the remote directory recursively before proceeding.
Field separator	Specifies the field separator used in the CSV file being imported. Default separator is , (comma).
Ignore CDR entries before	Specifies that CDR entries older than the specified date are not processed.
Lookup usernames in	Specifies where Smile will look to match usernames on CDRs. • **Subservices and Subscriptions** • **Subscriptions** • **Subservices**
Convert FNN Caller & Called numbers to E.164 format	When selected specifies that Australian Full National Number Caller and Called numbers are converted to E.164 international public telecommunication numbering format.
Convert usernames to E.164 format	When selected specifies that usernames are converted to E.164 format.
Age threshold to be *n* hours "recent"	Specifies the number of hours in the past of when files are still considered recent.
Error Ticket	Specifies if a helpdesk ticket is created by Smile in the event of an import error.

	• **Don't raise tickets on errors** • ***template ticket name*** For more information on template tickets, see the *Configuration Guide*.
Service	Specifies services that Smile will search to find matching subscriptions.
Manage Scheduled Task	Specifies when and how often the importer runs to automatically import CDRs. For more information on task scheduling, see the *Configuration Guide*.

JExpr field properties

The following fields are used to configure the JExpr properties of the flexible importer.

Skip expression

Values: true, false

Purpose: Specifies header and footer rows of the CSV file. When true the entry is skipped.

Example:

```
$1 != "CDR"
```

Column 1 is not equal to "CDR". Any rows that do not contain the text "CDR" in the column 1 cell will be skipped.

Item validity expression

Values: true, false

Purpose: Used to detect malformed rows. When false the entry is not valid

Example:

```
$$ == 4
```

There must be exactly 4 columns for an item row.

Username

Format: String

Purpose: Specifies the username to apply the CDR against.

Example:

```
$2
```

The username is the entry in column 2.

Call timestamp

Format: Java date/timestamp (java.util.Date)

Purpose: Specifies the date and time the CDR event began or occurred.

Example:

```
parsers.parseDate("yyyy-MM-dd HH:mm:ss", $11)
```

The entries in column 11 are currently formatted in the format yyyy-MM-dd HH:mm:ss. parseDate parsers the date to...

Caller number

Format: String

Purpose: Specifies the number from which the CDR event originated.

Example:

```
$6
```

The caller number is in the sixth column.

Called number

Format: String

Purpose: Specifies the number to which the CDR event was sent.

Example:

```
$7
```

The called number is in the seventh column.

CDR billable duration in seconds

Format: UTInteger

Purpose: Specifies the duration of the CDR. Must be > 0 to be billable.

Example:

```
$5
```

The CDR billable duration in seconds is in the fifth column.

Wholesale price

Format: UTDecimal

Purpose: Specifies the charge for the CDR from the vendor.

Example:

```
$3::UTDecimal
```

Convert the entry in column 3 to javaBigDecimal format.

Tariff code

Format: String

Purpose: Specifies the tariff code which Smile should use to rate the call.

Example:

```
$4 + "-" + $12
```

Concatenates the contents of column 4 with column 12, with an intervening hyphen.

Call type

Values: CallType.getVoice(), CallType.getData(), CallType.getSms(), CallType.getMms(), CallType.getFax(), CallType.getWap(), CallType.getForwardedCall(), CallType.getCount(), CallType.getUnknown(), CallType.getVideoCall(), CallType.getIsdn(), CallType.getImportedCharge()

Purpose: Specifies the call type. Defaults to CallType.getVoice()

Example:

```
CallType.getVoice()
```

The call type is voice.

Chargeable

Format: Boolean

Purpose: Specifies if the CDR entry is chargeable. When false Smile will not rate the call. Defaults to true.

Example:

```
false
```

The call is not chargeable.

Description of event

Format: Text

Purpose: Specifies a custom description of the call event. When set, Smile will by default separate the call's charge to its own line item with this description. Mostly useful for imported charges.

Example:

```
$13
```

The description of the event is in the thirteenth column.

Bytes uploaded by customer

Format: UTLong

Purpose: Specifies the amount of data uploaded in bytes.

Example:

```
$14::UTLong
```

The bytes uploaded by customer is in the fourteenth column.

Bytes downloaded by customer

Format: UTLong

Purpose: Specifies the amount of data downloaded in bytes.

Example:

```
$15::UTLong
```

The bytes downloaded by customer is in the fifteenth column.

Pages

Format: UTInteger

Purpose: Specifies the number of (fax) pages to be billed.

Example:

```
$16::UTInteger
```

The number of pages to bill is in the sixteenth column.

Count

Format: UTInteger

Purpose: Specifies the number of CDR events to be billed.

Example:

```
$17::UTInteger
```

The number of CDR events to bill is in the seventeenth column.

Percentile rate

Format: UTDecimal

Purpose: This figure is used to convert the single once-off activity this CDR represents (such as a text message or flat-rate phone call) into a quantity of billing units that are later translated into a retail price to bill the subscriber for.

Example:

```
$18::UTDecimal
```

The percentile rate is in the eighteenth column.

Bytes sent from customer percentile rate

Format: UTDecimal

Purpose: This figure is used to convert the amount of bytes transmitted on the network from this subscriber into a quantity of billing units that are later translated into a retail price to bill the subscriber for.

Example:

```
$19::UTDecimal
```

The bytes sent from customer percentile rate is in the nineteenth column.

Bytes sent to the customer percentile rate

Format: UTDecimal

Purpose: This figure is used to convert the amount of bytes received thru the network by this subscriber into a quantity of billing units that are later translated into a retail price to bill the subscriber for.

Example:

```
$20::UTDecimal
```

The bytes sent from customer percentile rate is in the twentieth column.

IP Address

Format: String

Purpose: Specifies the IP address of the client's device.

```
$21
```

The IP address is in the twenty-first column.

JExpr dictionaries

The JExpr language can feature pre-defined dictionaries for certain parts of Smile that accept JExpr expressions for their configuration.

Dictionaries are a list of pre-defined functions. These functions can take zero or more arguments, each argument being a JExpr expression itself. The function will return an expression of a particular JExpr type (or may return null).

A function call in JExpr looks like the following:

```
parsers.parseDate("yyyy-MM-dd", $1);
```

In this example:

- `parsers` is the dictionary namespace that contains the function
- `parseDate` is the name of the function
- This function takes two arguments (both of string type), and returns a Java date/timestamp (java.util.Date) value
- Parentheses encapsulate all of the arguments
- Commas separate individual arguments

parsers dictionary

This dictionary contains functions that direct how specified strings should be parsed.

`parseDate()`

```
parsers.parseDate(format, sourceText)
```

Parses a string containing a date and/or timestamp specification in a particular structured format.

Parameters

format

Format: string

Purpose: Describes the expected layout of the individual date/time components within the sourceText argument.

The format string can contain the following (case-sensitive) placeholders:

yy	Two-digit year. For example, `99` for 1999.
yyyy	Four-digit year. For example, `1999`.
MMMM	Full month name. For example, `July`.
MMM	Three-letter month abbreviation. For example, `Jul` for July.
MM	Two-digit numeric month. For example, `07` for July.
dd	Two-digit day in month. For example, `23`.
HH	Two-digit hour-in-day. For example, `00` thru `23` inclusive, for 24-hour time.
hh	Two-digit hour-in-day. For example, `01` thru `12` inclusive, for 12-hour time.
mm	Two-digit minute in hour. For example, `00` thru `59` inclusive.
ss	Two-digit second in minute. For example, `00` thru `59` inclusive.
SSS	Three-digit millisecond value. For example, `000` thru `999` inclusive.
z	Time zone. For example, `UTC+10:00`.
Z	RFC 822 time zone. For example, `+1000` for 10 hours ahead of UTC.
a	AM/PM marker. For example, `AM` or `PM`.

To escape a letter (indicate that it needs to be matched verbatim), enclose it in single quotes `'`. To match a single quote directly, write two single quotes consecutively `''` (no space between quotes).

Note: All the sequences supported by the Java Foundation Class java.text.SimpleDateFormat are permitted for this parameter. For more information, see http://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

The letters 'G', 'Y', 'L', 'W', 'w', 'D', 'F', 'E', 'u', 'k', 'K', 'X' are also supported as per the above Java specification.

Any other characters, such as punctuation, spaces or letters not mentioned above, will be matched verbatim.

sourceText

Format: string

Purpose: Specifies the date string to be parsed.

Example: A parseDate request

The following example is a `parseDate` request containing a string for the `format` argument and `sourceText` of the date to parse.

```
parsers.parseDate("dd/MM/yyyy HH:mm:ss", "31/12/1999 23:59:59")
```

Result

Returns a java.util.Date instance containing the parsed date/timestamp.

Example: A returned parseDate request

This example shows the returned parseDate request.

```
31st December 1999 at 11:59:59pm
```

CallType dictionary

All the functions in this dictionary return a specific value from a set of enumerated values that can be used to categorise a type of CDR record.

None of these functions take any arguments.

`getVoice()`

```
CallType.getVoice()
```

Returns ordinary voice call values.

`getData()`

```
CallType.getData()
```

Returns data session values.

`getSms()`

```
CallType.getSms()
```

Returns SMS text message values.

`getMms()`

```
CallType.getMms()
```

Returns MMS message values.

`getFax()`

```
CallType.getFax()
```

Returns fax call values.

`getWap()`

```
CallType.getWap()
```

Returns WAP session values.

`getForwardedVoice()`

```
CallType.getForwardedVoice()
```

Returns forwarded voice call values.

`getCount()`

```
CallType.getCount()
```

Returns CDR total count record values.

getUnknown()

```
CallType.getUnknown()
```

Returns unknown call type values.

getVideoCall()

```
CallType.getVideoCall()
```

Returns video call values.

getIsdn()

```
CallType.getIsdn()
```

Returns ISDN session values.

getImportedCharge()

```
CallType.getImportedCharge()
```

Returns imported charge (from external system) values.

importerContext dictionary

This dictionary contains functions that return information about the current input file.

getFilename()

```
importerContext.getFilename()
```

Returns the file name of the file currently being processed.

Parameters

No parameters.

Returns

Returns a string value.