IBM Safer Payments Manual

This manual describes the IBM Safer Payments software product. It has been generated automatically from within the software itself.

Contents

1. Introduction

This chapter contains all you need to quickly get familiar with IBM Safer Payments.

back to top

1.1 Quick Facts

Some basics to introduce IBM Safer Payments:

What is IBM Safer Payments?

- IBM Safer Payments prevents fraud in cashless payment systems – such as credit cards – by analyzing transaction streams and intercepting high-risk transactions before they are completed.
- To detect fraud in transaction patterns and to differentiate fraudulent from legitimate payment behaviour, IBM Safer Payments analyses transaction sequences and uses "if-then" rules to conclude for each transaction whether or not to recommend authorization.
- IBM Safer Payments is a software program that runs at the authorization center of banks and payment processors, or at switches. It is tied to the existing authorization and card management systems and payment gateways.

What kind of software is it?

- IBM Safer Payments runs on Red Hat Linux.
- IBM Safer Payments runs as a "server" component (daemon process).
- IBM Safer Payments is self-contained. Its database and application server are embedded. To run IBM Safer Payments, you need hardware and an operating system.
- The IBM Safer Payments user access interface is any standard web browser supporting JavaScript.
- Other systems connect to IBM Safer Payments either via IP messages (real-time) or batch files.
- For high-availability needs, IBM Safer Payments can be installed in a clustered configuration.

What is the architecture of the IBM Safer Payments software?

IBM Safer Payments combines the advantages of a number of proven IT architecture patterns:

- Internally IBM Safer Payments is based on a service oriented architecture where each task to be performed is spun off as a separate service. Some of these services are further split into multiple threads to support parallel computation.
- Interfacing with other systems, IBM Safer Payments supports a service-provider / service-consumer type interface pattern (similar to the SOAP standard).
- Interfacing to users, IBM Safer Payments supports a lightweight, JavaScript/AJAX based browser client that is based on the Model View Controller architecture pattern.

What is there to an IBM Safer Payments installation?

- The IBM Safer Payments main server component is installed on a server computer.
- The IBM Safer Payments client does not require any installation.

back to top

1.2 IBM Safer Payments User Access

IBM Safer Payments user access is designed for productivity. While it follows general accepted look&feel and usage metaphors of web technology based user interaction, there are a few "extras" that IBM Safer Payments features, which can greatly enhance productivity.

Top horizontal navigation bar

IBM Safer Payments is structured in five main functions:

- Dashboard
  Quick overview on IBM Safer Payments vital functions.
- Report
  Define and execute IBM Safer Payments standard reports.
- Investigation
  Working cases and querying data.
- Model
  Drill-down analysis of data and IBM Safer Payments performance. Creation of model revisions.
- Administration
  Configuration, settings, and administration functionality.

These functions can directly be navigated to by clicking on one of the five tabs located at the left of the top navigation bar. Most functions, once selected, display a second vertical navigation menu on the left side of the page. Navigation choices are highlighted when the mouse pointer moves over them.

There are three more functions displayed as tabs on the right side of the top navigation bar:

- New

Opens a new tab/window. See below for details.

- Help
  Help on specific topics and generation of full IBM Safer Payments user manual.
- Logout
  Ends a session. Notice that while IBM Safer Payments logs you out after a certain period of inactivity, some functions, such as automatic rule generation will keep a session alive indefinitively. It is thus important to use the logout function once you completed your work in IBM Safer Payments.

Multiple windows

For certain actions, you may prefer to view multiple aspects of IBM Safer Payments, or work on multiple elements at once. For this, IBM Safer Payments supports opening any number of tabs or browser windows on your computer, sharing the same user session. To facilitate this process, the top navigation bar offers the "New" tab that opens another browser tab or window (depending on your browser settings) using the same session. IBM Safer Payments is designed that you may work in multiple tabs/windows in parallel.

Feedback

Most user actions that are executed on the IBM Safer Payments server return a "feedback status". The statuses are:

- ✓ OK
  Confirmations of successfully executed actions are indicated by a short (green) text in the rightmost corner of the web page above the horizontal navigation bar.
- ⚠ Warning
  Warnings are indicated by a short (orange) text in the rightmost corner of the web page above the horizontal navigation bar. IBM Safer Payments issues warnings when it needs to alert the user to circumstances relating to the action executed, even though the action had been executed.
- ● Error
  If the action could not be executed for reasons the user can change, IBM Safer Payments responds with a dialog explaining the error. The dialog must be closed before the user can continue.
- ● Fatal Error
  The action could also not be executed, but IBM Safer Payments considers that the user cannot change the circumstances for this error to occur. In most cases, a fatal error indicates a software related problem that can only be resolved with assistance from the manufacturer of IBM Safer Payments.

Notice that the "OK" and "warning" feedbacks are only displayed for a few seconds. To see the last feedbacks, just click right at the position the feedbacks usually appear to open a list of the past feedbacks.

Sections and toolbars

Each IBM Safer Payments page features one or more so-called sections. A section is a rounded-edge frame with a title/toolbar tab. The toolbar icons display a rectangular frame once the mouse pointer hovers over it to visualize when they are in focus. After stopping the mouse pointer over a toolbar icon, a tooltip-style pop-up explanation text of the toolbar action appears.

Some toolbar actions are "one-off" triggers of a certain activity (for instance, "save" actions); others are "toggles" and have an activated and a deactivated state. The active state of a toggle is indicated by a "green light" type indicator within the icon.

Tables

The IBM Safer Payments data tables are used on nearly every IBM Safer Payments page. They have a number of productivity features:

1. The width of each column can be changed by dragging the separator between the column headers. Note this behaviour with tables that display a two-line icon when hovering over the header borders.
2. The sequence of columns in a table (for most tables) can be changed by dragging columns (header field) with the mouse. A red insert marker indicates the "drop zone" for the column.
3. Each table has a maximum row size. If there are fewer rows than this size, the table is displayed in full. Once the number of table rows exceeds the number of maximum rows, the table is displayed in "paged" mode, where page selectors automatically appear at the bottom of the table. In paged mode, the maximum row size can be adapted by dragging the horizontal line at the lower frame of the table.
4. You may sort the table by any column by simply clicking on the column head. Clicking again toggles between ascending and descending order. Multi-criteria sort is accomplished by subsequentially sorting different columns. To reset sort settings, pagination, and column width and sequence, click right on the column header and select "reset table preferences" from the pop-up menu.
5. In most tables, you may highlight rows using a single left mouse click. In some tables, you may highlight multiple consecutive rows by holding the [Shift] key pressed and then clicking the start and end row subsequently. Non-consecutive rows can be highlighted by holding the [Ctrl] key and clicking the individual rows. Depending on the table, a right click on one of the highlighted rows opens a context menu that provides direct access to possible operations on the element represented in this row. On certain tables, the actions for multiple vs. single highlighted rows are different.

All settings listed above are stored as "preferences" with the user account. Once you get back to the same table, the last settings you have chosen are restored. Notice that you may reset all preferences at once from the "my account page" and that you may reset the preferences of just one table by clicking with the right mouse button on a column header.

Keyboard shortcuts/accelerators

While IBM Safer Payments in using a web browser interface that is optimized for mouse usage, IBM Safer Payments also offers a number of keyboard shortcuts. Depending on the browser, shortcuts use the [Ctrl] key held and then a keyboard letter pressed and/or the [Ctrl]-[Alt] key combination held (press both) and a letter key:

- On most pages displaying a table of elements, the keyboard shortcut [Ctrl]-[N] creates a new element. (When using the Chrome-Browser the shortcut variant [Ctrl]-[Alt]-[N] has to be used)
- On most pages displaying an entry form, the keyboard shortcut [Ctrl]-[S] saves the contents of the form and closes the form.
- On most pages displaying an entry form, the keyboard shortcut [Ctrl]-[D] deletes the opened element (confirmation dialog).
- On most pages displaying a condition/conclusion entry form, the keyboard shortcut [Ctrl]-[Shift]-[I] creates a new condition/conclusion as long as the focus is on any entry field of the condition form .

Notice that you may use the tab key and shift-tab key combination to navigate the fields of a form.

Useful hints

- The IBM Safer Payments user interface uses both modal and modeless "dialog boxes" (windows that open within the browser window). A modal dialog box requires the user to supply information, or cancel the dialog box, before allowing the application to continue. A modeless dialog box allows the user to supply information and return to the previous task without closing the dialog box. IBM Safer Payments uses modeless dialog boxes for instances with hyperlink query results and for exploring attribute values in counter/rule generation. Notice that a quick way to close such dialogs is to use the [Esc] key on your keyboard.

Remarks

- If for whatever reasons, the UI behaves "funny", refresh the page (e.g. hit the [F5] key). Refreshing causes a complete restart and reload of the UI without losing the user session. You may, however, lose data entered on the form that is not yet saved.

back to top

2. Ticker

This chapter provides information about the ticker.

back to top

2.1 Ticker

The ticker is a fast and easy way to inform colleagues about important events. It displays text messages that were created by other users - so called ticker entries - and offers the possibility to write text messages yourself.

Every ticker entry has at least a title, but can also contain additional information within a note.

Ticker entries are assigned to a mandator. All users having the ticker view privilege for the respective mandator are able to view this entry. Users having the delete privilege for the respective mandator, are able to delete ticker entries. In addition to the belonging mandator, submandators can be selected from the "Display in" menu when creating an entry. Users that do not have the ticker view privilege for the ticker entry's mandator, but for one of the "Display in" mandators, are able to view the entry (but cannot delete it). This allows you to address colleagues of submandators, too.

Saved ticker entries are sorted chronologically (new entries on top). Entries that were added since your last log out are displayed in bold until you close the ticker. Ticker entries that were created recently are displayed in bold, too. The settings for the time a ticker entry is considered as new and thus printed in bold can be changed in "My Account" ("Ticker entry highlight interval"). To load new ticker entries that were created since you opened the ticker you can refresh the ticker manually by clicking on its refresh button. In "My account" there is a setting for an automatic refresh interval as well ("Ticker refresh interval").

The ticker can be opened and closed at any time. It is possible to open multiple tickers at a time by clicking the tab several times.

Once deleted, ticker entries cannot be restored. Deleting a ticker entry will not only delete the ticker entry for yourself but for all other users as well. If a certain number of overall ticker entries is reached, the oldest entry will be deleted. This number can be defined in system configuration.

As an example use the ticker to inform about new CPPs. If the ticker title and a CPP name are identical, there will be a hyperlink to the list of cases connected with the CPP. You can insert the name of a CPP manually or use the functionality within the CPP table context menu to add the name.

back to top

2.2 New ticker entry

Create a new ticker entry by filling in this form. Click save to add it to the ticker.

- **Title**
  This is the title of the ticker entry. Use it as only message of the entry or add a note.
- **Note**
  Additional information can be added here.
- **Mandator**
  This is the mandator the ticker entry you are creating will belong to. Users with the ticker view/delete privilege for this mandator

are able to view or delete this entry.

- **Display in**
  If you want the entry to be visible in submandators of the chosen mandator additionally, you can add these submandators here. Users of this submandators can view the entry but cannot delete it.

back to top

3. Dashboard

The IBM Safer Payments dashboard provides a quick overview on IBM Safer Payments operations.

back to top

3.1 Status Alarm Indicators

Status alarm indicators (SAI) constantly monitor operational parameters. The traffic light coding, green for "OK", yellow for "warning", and red for "error" in particular provide a one-glance overview on IBM Safer Payments' operational health.

SAI are configured individually for each mandator. Their display on the dashboard page consists of the colour icon and a brief explanation text, typically containing the value of the SAI. Typically an extended explanation text with more data is available as tooltip when the mouse pointer rests over the short text. Depending on the SAI's configuration, a state change from "OK" to "warning" or "error" may have triggered the generation of an email or mobile text.

Multiple IBM Safer Payments instance installation

Most IBM Safer Payments installations involve multiple IBM Safer Payments instances in a cluster. Since in such a setup, only one IBM Safer Payments instance serves user access, SAI for all instances are accessible from each of the IBM Safer Payments instances.

Notice that there are SAI that monitor parameters that are the same on all instances (e.g. "open cases"), while there are others (e.g. "transaction messages processed") that are different for each instance. For the latter, a separate SAI is shown for each IBM Safer Payments instance. The ID of the respective instance is shown in square brackets with the brief explanation text.

Filter

The "filter" drop list box located in the header lets you restrict the SAI shown:

- All
  Shows SAI of any status and any cluster instance.
- Errors
  Shows only SAI of "error" status for all cluster instances.
- Warnings
  Shows only SAI of "warning" status for all cluster instances.
- W + E
  Shows SAI of "error" and "warning" status for all cluster instances.
- IBM Safer Payments *n*
  Shows SAI of any status only for cluster instance *n*.

Remarks

- Each SAI can only be a "warning" or an "error". If for a certain parameter, both warning and error (with different thresholds) shall be displayed, they are defined as two separate SAI.

back to top

3.2 Key Performance Indicators Online Help

Key performance indicators (KPI) constantly supervise operational performance and plot them over time in one or more charts.

KPI are configured individually for each mandator. Their display on the dashboard page consists of a curve and an entry in the legend right of the respective chart. The exact values can be viewed when the mouse pointer rests over the chart. In this case, the chart shows a red vertical line as time cursor, and the exact values of the KPIs are shown in the legend. Notice that if there are different time points for the KPIs, the legend value shows the data point nearest to the cursor.

Multiple IBM Safer Payments instance installation

Most IBM Safer Payments installations involve multiple IBM Safer Payments instances in a cluster. Since in such a setup, only one IBM Safer Payments instance serves user access, KPIs for all instances are accessible from each of the IBM Safer Payments instances.

Notice that there are KPIs that monitor parameters that are the same on all instances (e.g. "open cases"), while there are others (e.g. "transaction messages processed") that are different for each instance. For the latter, a separate KPI is shown for each IBM Safer Payments instance. The ID of the respective instance is shown in square brackets with the legend text.

back to top

4. Report

This chapter covers the report function of IBM Safer Payments.

back to top

4.1 Case Class Reports

The table below lists all defined case class reports. Case class reports summarise case investigation activities on a per case class basis.

back to top

4.1.1 Case Class Report Definition

Case class reports have the following settings:

- **Enabled**
  Lets you temporarily display or hide reports.
- **Name**
  Name of report
- **Comment**
  Description of report
- **Mandator**
  Each report belongs to one mandator.
- **Use data from**
  Lets you select which data source to use in case that one mandator has several submandators.
- **Data range**
  Lets you specify the reporting time period.
- **Case Conditions**
  You may further restrict the cases to those whose reporting attributes and data satisfy certain criteria by defining conditions here. If no conditions are defined, all cases satisfying the general settings (above) are included in the case class report.

back to top

4.1.2 Case Class Report

The case class reports provides a comparison of the individual case class performance with respect to a number of key performance indicators (KPIs). While delivering basis data to optimise case investigation operations, it also provides valuable feedback to the fraud analysts that maintain the rules generating cases. The report lists a row for every case class for which cases are found in IBM Safer Payments. For each case class, the columns provide different types of information:

Current state

The columns summarise for each case class how many cases currently exist in IBM Safer Payments that were generated within the reporting period in the respective case state.

Summary

In addition to case state statistics, the following information is provided:

- **Due cases**
  Cases that have been postponed/forwarded that are due as of now.
- **Bulk transitioned cases**
  Cases that were last processed using the bulk transition function.
- **Not closed**
  Cases that have not been closed (sum of cases that are not closed).
- **Total cases**
  Total of cases in IBM Safer Payments in reporting period (sum of cases in all states).

Created cases

Number of cases that were generated during the reporting period in this case class (if the case class changed because of alarm aggregation the case is reported in the last assigned case class).

Closed cases

Number of cases that were closed during the reporting period (regardless of their generation date). The numbers are shown according to sub-criteria and total columns:

- **Fraud**
  Number of cases closed with a close code classified as "fraudulent"
- **Genuine**
  Number of cases closed with a close code classified as "genuine"
- **Unknown**
  Number of cases closed with a close code classified as "unknown.
- **Total closed**
  Total number of cases closed (sum of the three previous columns)

Key Performance Indicators

The columns provide key performance indicators for fraud prevention performance on cases generated during the reporting period:

- **Investigation rate**
  Ratio of cases that could be closed with a close code classified as either "fraudulent" or "genuine" (not "unknown"). A high investigation rate indicates that investigators can successfully determine whether or not generated cases are fraudulent.
- **Hit rate**
  Ratio of cases that have been closed with a close code classified as "fraudulent" versus all cases closed. It thus states what fraction of the cases generated actually turned out to be fraud.
- **False positives**
  Ratio of cases that could be closed with a close code classified as "genuine" divided by the number of cases closed as "fraudulent".

Timing

The columns provide key statistical benchmark data on the response speed of investigation operations (based on cases generated during the reporting period):

- **Lead**
  Average time between case generation and first investigation (indicator of how quickly investigators picks up on cases).
- **Work**
  Average time between first investigation of a case and its closing
- **Open**
  Average time between generation of a case and its closing
- **Open fraud**
  is the same indicator as "open" but only considers cases that were later identified as "fraudulent".

back to top

4.2 Investigation Reports

The table below lists all defined investigation reports. Investigation reports summarise case investigation activities on a daily or hourly basis, respectively.
back to top

4.2.1 Investigation Report Definition

Investigation reports have the following settings:

- **Enabled**
  Lets you temporarily display or hide reports.
- **Name**
  Name of report
- **Comment**
  Description of report
- **Mandator**
  Each report belongs to one mandator.
- **Use data from**
  Lets you select which data source to use in case that one mandator has several submandators.
- **Type**
  Lets you select which type of report to be generated:
  - Investigation daily/hourly report
    This report provides an overview on case generation and case investigation (closing) performance. Depending on the selected type IBM Safer Payments shows one line per hour or one line per day.
- **Reference parameter**
  Lets you specify whether the results should refer to generated cases or to closed cases.
- **Data range**
  Lets you specify the reporting time period.

- **Case Conditions**
  You may further restrict the cases to those whose reporting attributes and data satisfy certain criteria by defining conditions here. If no conditions are defined, all cases satisfying the general settings (above) are included in the case class report.

back to top

4.2.2 Daily Investigation Report

The daily investigation report provides a quick overview on daily operations for all calendar days in the report period defined. Each day of the reporting time period is printed as a row of the report. For each day, the following information is provided:

Created cases

Number of cases that were generated on that date

First investigated cases

Number of cases that were first investigated on that date

Not closed cases

Number of cases that were generated on that date but *not* yet closed as of now. For each case state, the number of cases is shown in the respective column. In addition to case state statistics, the last column presents summary of the previous columns.

Closed cases

Number of cases that were generated on that date and are closed as of now. The numbers are shown according to sub-criteria and total columns:

- **Fraud**
  Number of cases closed with a close code classified as "fraudulent"
- **Genuine**
  Number of cases closed with a close code classified as "genuine"
- **Unknown**
  Number of cases closed with a close code classified as "unknown"
- **Total**
  Total number of cases closed (sum of the previous three columns)
- **Total bulk closed**
  Total of cases closed using bulk close functionality (part of the total sum, includes all close code classifications)

Key Performance Indicators

The columns provide key performance indicators on fraud prevention performance on cases generated on date:

- **Investigation rate**
  Ratio of cases that could be closed with a close code classified as either "fraudulent" or "genuine" (not "unknown"). A high investigation rate indicates that investigators can successfully determine whether or not generated cases are fraudulent.
- **Hit rate**
  Ratio of cases that was closed with a close code classified as "fraudulent" versus all cases closed. It thus states what fraction of the cases generated actually turned out to be fraud.
- **False positives**
  Ratio of cases that could be closed with a close code classified as "genuine" divided by the number of cases closed as "fraudulent".

Timing

The columns provide key statistical benchmark data on the response speed of investigation operations (based on cases generated on date):

- **Lead**
  Average time between case generation and first investigation (indicator of how quickly investigators picks up on cases)
- **Work**
  Average time between first investigation of a case and its closing
- **Open**
  Average time between generation of a case and its closing
- **Open fraud**
  is the same indicator as "open" but only considers cases that were later identified as "fraudulent"

back to top

4.2.3 Hourly Investigation Report

The hourly investigation report provides a quick overview on daily operations for all full hours in the report period defined. Each hour of the reporting time period is printed as a row of the report. For each hour, the following information is provided:

Created cases

Number of cases that were generated during that hour on that date.

First investigated cases

Number of cases that were first investigated during that hour on that date.

Not closed cases

Number of cases that were generated during that hour but *not* yet closed as of now. For each case state, the number of cases is shown in the respective column. In addition to case state statistics, the last column presents summary of the previous columns.

Closed cases

Number of cases that were generated during that hour on that date and are closed as of now. The numbers are shown according to sub-criteria and total columns:

- **Fraud**
  Number of cases closed with a close code classified as "fraudulent".
- **Genuine**
  Number of cases closed with a close code classified as "genuine".
- **Unknown**
  Number of cases closed with a close code classified as "unknown".
- **Total**
  Total number of cases closed (sum of the previous three columns).
- **Total bulk closed**
  Total of cases closed using bulk close functionality (part of the total sum, includes all close code classifications).

Key Performance Indicators

The columns provide key performance indicators on fraud prevention performance on cases generated during that hour on that date:

- **Investigation rate**
  Ratio of cases that could be closed with a close code classified as either "fraudulent" or "genuine" (not "unknown"). A high investigation rate indicates that investigators can successfully determine whether or not generated cases are fraudulent.
- **Hit rate**
  Ratio of cases that were closed with a close code classified as "fraudulent" versus "genuine" or "unknown". It thus states what fraction of the cases generated actually turned out to be fraud.
- **False positives**
  Ratio of cases that could be closed with a close code classified as "genuine" divided by the number of cases closed as "fraudulent".

Timing

The columns provide key statistical benchmark data on the response speed of investigation operations (based on cases generated during that hour on that date):

- **Lead**
  Average time between case generation and first investigation (indicator of how quickly investigation picks up on cases).
- **Work**
  Average time between a case was first investigated and its closing.
- **Open**
  Average time between a case was generated and its closing.
- **Open fraud**
  is the same indicator as "open" but only considers cases that later are identified as "fraudulent".

back to top

4.3 Missed Cases Reports

The table lists all missed cases reports that are defined and for which you have access privileges. A missed cases report provides information about wrong decisions when closing a case.

The report will return a case if the following is fulfilled:

The case class of the case has an aggregation attribute that is used in an index with a sequence, for example the aggregation attribute is PAN and there is an account index defined on PAN which has a sequence. The case was closed as genuine or unknown, for example on 2021-05-01 12:00:00. There exists at least one fraud transaction in the sequence of the index and the value of the meta attribute timestamp of this fraudulent transaction is within the defined time horizon when this time is compared to the close date of the case. For example, if the transaction's timestamp is 2021-05-01 18:00:00 and the horizon is defined as 10 hours before and after, this would return the case that was closed on 2021-05-01 12:00:00. If the fraudulent transaction would have a timestamp 2021-05-01 23:00:00, the case would not be returned as this is not within the time horizon.

The reference parameter plus data range lets you define which cases should potentially be included in the report. For example, if you choose “generated cases” it will only return cases generated during the defined data range.

back to top

4.3.1 Missed Cases Report Definition

Missed cases reports have the following settings:

- **Enabled**
  Lets you temporarily display or hide reports.
- **Name**
  Name of report
- **Comment**
  Description of report
- **Mandator**
  Each report belongs to one mandator.
- **Use data from**
  Lets you select which data source to use in case that one mandator has several submandators.
- **Reference parameter**
  Lets you specify whether the results should refer to generated cases or to closed cases.
- **Data range**
  Lets you specify the reporting time period.
- **Case Classes**
  Only cases from the selected case classes will be included in this report.
- **User accounts**
  Only cases closed by the selected user accounts will be included in this report.
- **Time horizon before**
  A case is considered "missed fraud" (and thus included in this report), if a fraudulent transaction occurred within this time period before it was closed "genuine" or "unknown".
- **Time horizon after**
  A case is considered "missed fraud" (and thus included in this report), if a fraudulent transaction occurred within this time period after it was closed "genuine" or "unknown".
- **Report Attribute Conditions**
  You may further restrict the cases to those whose reporting attributes satisfy certain criteria by defining conditions here. If no conditions are defined, all cases satisfying the general settings (above) are included in the case class report.

back to top

4.3.2 Missed Cases Report

The missed cases reports provides information about wrong decisions when closing a case. In order to learn from such wrong decisions the report lists all the cases that had been closed as "genuine" or "unknown" but fraud is later on reported on the account. The columns provide the following information:

- **Case ID**
  Unique ID of a case
- **Case Class**
  Case Class where the case was closed from
- **Fraud Status**
  Status the case was closed with like "genuine" or "unknown"
- **Closed on**
  Date when case was closed
- **Closed by**
  User who closed the case
- **Fraud on**
  Date of fraud

Each line refers to one case and a hyperlink function leads to the appropriate case details.

back to top

4.4 Investigator Reports

The table lists all investigator reports that are defined and for which you have access privileges. Investigator reports summarise case investigation activities on a per user and per case class basis.

back to top

4.4.1 Investigator Report Definition

Investigator reports have the following settings:

- **Enabled**
  Lets you temporarily display or hide reports.
- **Name**
  Name of report
- **Comment**
  Description of report
- **Mandator**
  Each report belongs to one mandator.
- **Use data from**
  Lets you select which data source to use in case that one mandator has several submandators.
- **Users**
  Lets you restrict the report to a certain amount of users being displayed.
- **Case Classes**
  Lets you restrict the report to a certain amount of case classes being displayed.
- **Data range**
  Lets you specify the reporting time period.
- **Case Conditions**
  You may further restrict the cases to those whose reporting attributes and data satisfy certain criteria by defining conditions here. If no conditions are defined, all cases satisfying the general settings (above) are included in the case class report.

back to top

4.4.2 Investigator Report

The investigator user report provides a comparison of the individual performance of investigators. The report lists each user with an investigation privilege as one row. For a user that had activity with cases generated in the time period, a subsequent row is shown for any case class that had activity. These rows are indented and the case class name is displayed in the left column to provide a detailed breakdown of a user's activity. For each user, different types of information are provided:

Investigation actions

Number of investigation actions per case state taken by the user within the reporting period. For the case class detail rows, if the case class has changed because of alarm aggregation, the case is reported in the last assigned case class. Notice that each action is counted separately, that is, if a user opened a case in one state, and changed the state by executing a transition, these actions will be reflected in both state columns.

Investigation results

The columns detail the user's performance within the reporting period for closed cases by the categories:

- **Fraud**
  Number of cases closed as "fraudulent"
- **Genuine**
  Number of cases closed as "genuine"
- **Unknown**
  Number of cases closed as "unknown"

Key performance indicators

The columns provide key performance indicators derived from the investigation results (shown in the columns to the left):

- **Investigation rate**
  Ratio of cases that could be closed with a close code classified as either "fraudulent" or "genuine" (not "unknown"). A high investigation rate indicates that investigator could successfully determine whether or not generated cases are fraudulent.
- **Hit rate**
  Ratio of cases that was closed with a close code classified as "fraudulent" versus "genuine" or "unknown". It thus states what fraction of the cases generated actually turned out to be fraud.
- **False positives**
  Ratio of cases that could be closed with a close code classified as "genuine" divided by the number of cases closed as "fraudulent".

back to top

4.5 Message Report

The table lists the volume of all messages of the different MTIDs by the time period in which they were processed by IBM Safer Payments. It contains a column for each MTID that was processed, and a row for each time period.

Integration with IBM License Metric Tool

IBM Safer Payments generates IBM Software License Metric Tag (SLMT) files. Versions of IBM License Metric Tool that support IBM Software License Metric Tag can generate License Consumption Reports. Read this section to interpret these reports for IBM Safer Payments.

Each instance of a running IBM Safer Payments environment generates an IBM Software License Metric Tag file. As there is no master instance, the same SLMT file is created for each instance.

Customer has to select one SLMT file and ignore the others, as the data provided in there is identical and each file covers the whole environment.

The metric type monitored is RVU. The value is refreshed at least once a day within the end of day job.

About the RVU metric

The metric RVU has different subtypes.

- **Subtype transactions**
  The value reported for this metric is the number of all transactions of all instances per month. The counter resets at the beginning of every month. This means, that even the time period of a single SLMT file is just one day, transactions are counted from the beginning of the month.
- **Subtype accounts**
  This subtype is currently not measured with IBM Safer Payments and therefore no SLMT file is created for accounts.

The IBM Software License Metric Tag file is in the /rep directory of each instance configuration.

back to top

5. Investigation

This chapter covers the investigation function of IBM Safer Payments.

back to top

5.1 Case Investigation Workflow

This chapter covers the case investigation workflow of IBM Safer Payments.

back to top

5.1.1 Case Selection

This section enables a quick definition of selection criteria for investigation cases to be displayed in the section below.

- **Mandators**
  Defines which mandators' cases are to be displayed. This selection is only shown when you have the privileges to see cases from multiple mandators.
- **Case classes**
  Defines which case classes' cases are to be displayed. This selection is only shown when you have the privileges to see cases from multiple case classes.
- **Generation date**
  Restricts investigation cases shown to a specific generation date (time) interval. Leaving an entry field empty implies no restriction on the generation date.
- **Case states**
  Restricts display of cases to certain states.
- **Investigators**
  Defines which investigators' cases are to be displayed. Notice that not assigned cases are always displayed. To change this, use the case state selection criteria. Notice that this checkbox only appears if more than one user is defined for this mandator.
- **Working queues**
  Restricts display of cases to certain working queues. Cases that are not associated with any working queue are always displayed. You can uncheck all working queues to see not associated cases. Notice that this selection criteria only appears if there are working queues defined for this mandator. Working queues are defined on the page *Administration -> Case management -> Working queues*.
- **Case score**
  Restricts investigation cases shown to a specific case score interval. Leaving an entry field empty implies no restriction on the case score.
- **Case selection conditions**
  Restricts investigation cases shown to all cases that satisfy the defined conditions. All reporting attributes that belong to the selected case classes and case variables can be used in case selection conditions.

Remarks

- Notice that with each choice, the investigation case table shown in the section below is reloaded with the updated cases.
- Depending on the "auto-refresh" settings ("administration" tab, "system configuration" page), this page reloads periodically to

update with potentially new cases.
- For all selections in this section, you will be presented either with a check box list or a drop down multi-select box, depending on the number of choices. You can define the number of choices above which the drop down multi-select box is shown rather than the check box list as user preference on the "my account" page.
- If you change the case class selection all previous defined conditions are removed.

back to top

5.1.2 Investigation Cases

The table shows all cases according to the selection criteria.

The following columns are always shown:

- **Case ID**
  This is an internal identification number in IBM Safer Payments. Each case will be assigned a unique value, a higher number always indicates a later generation, although the numbers are not sequential. The main use of this number is to provide a unique identification of a case.
- **Case class**
  Name of the case class to which this case belongs.
- **Mandator**
  Name of the mandator to which this case belongs (if there are multiple mandators).
- **Working queue**
  Name of the working queue this case is associated with.
- **Investigator**
  Name of the investigating user to whom this case is assigned.
- **Generated on**
  System timestamp of first alarm that is consolidated in this case.
- **Last action on**
  System timestamp of the last action performed to this case.
- **Case score**
  Highest case score meta attribute value of the alarms consolidated in this case.
- **Hits**
  Number of alarms consolidated in this case.
- **Case state**
  Name of the current state of the case as defined per case workflow settings.
- **Last case state**
  Name of the last state of the case.
- **Memo**
  Free text memo containing information about the case which is edited by the investigating user.
- **Case close code**
  For cases that has been closed this column shows the case close code which was used by the fraud investigator. Case close codes are defined on the page *Administration -> Case close codes*.
- **Fraud status**
  Each case close code is mapped to one of the principal fraud statuses "fraudulent", "genuine", or "unknown". When a case is closed the respective fraud status of the case close code is shown in this column.

The remaining columns are the reporting attribute values of all case classes to which cases in this table belong.

To work on a case, you can either double click on a row to open a full investigation page, or you can click the right mouse button on a row to list all actions for which you have privilege to execute. Depending on your privilages, the following actions are available:

- **Investigator actions**
  - **View**
    Same as double clicking, but opens the investigation page in "view-only" mode.
  - **Investigate case**
    Opens the investigation page (same as double clicking).
- **Supervisor actions**
  - **Take over case**
    Takes over a case that is currently in a follow up queue for a different user. This action can be executed on a single case at a time.
  - **Interrupt case**
    Interrupts a case that is currently being investigated by a different user. Interrupted cases are taken from their investigators and automatically set to state "New". This action can be executed on multiple cases at once.
  - **Execute case transitions**
    You may click on rows using the pressed [Shift] or [Ctrl] keys to select multiple rows individually or sequentially. When multiple rows are selected, you may right click on one of the selected rows to list available bulk transitions that can be executed on all selected cases in one step.

Remarks

- Notice that you can sort the investigation table by clicking on column headers. To sort for more than one column, simply click the columns in sequence (the former "inner" sorting will remain). Sorting preferences are stored with your user's account.
- The maximum number of cases shown on this table is limited by the general IBM Safer Payments setting. You can change this setting from the system configuration page.

back to top

5.1.3 Case Search

The case search page lets you quickly find a case based on the value of one of its reporting attributes. Simply select the reporting attribute from the drop down menu, and enter the value in the entry field right of the drop down menu. Formatting characters like digit group separator and decimal separators will be removed automatically. The only exception here is when you are searching by case id. The hyphen in case ids is mandatory.

back to top

5.1.4 My Working Queues

This section provides statistical information about cases of the selected working queues.

The following information is provided:

- **Number of cases**
  Number of cases contained in the selected working queues.
- **Oldest open case**
  The age of the oldest case in days contained in the selected working queues.
- **Average age**
  Average age of cases in days contained in the selected working queues.
- **Included case classes**
  Names of different case classes whose cases are included in the selected working queues.
- **Included case states**
  Names of different case states that are included in the selected working queues.

back to top

5.2 CPPs

The table shows all CPPs according to the selection criteria defined in the selection above.

The following columns are always shown:

- **Case group**
  Name of the case group the CPP belongs to.
- **Mandator**
  Name of the mandator to which this CPP belongs (Mandator is determined by case group).
- **Name**
  Name of the CPP.
- **Comment**
  Comment of the CPP.
- **Status**
  Current status of the CPP:
  - New
    CPP has not been opened.
  - Investigated
    CPP is currently being worked on.
  - Follow up
    CPP has been postponed for later follow up.
  - Due
    CPP has been postponed for follow up and is due to be re-opened now, or is overdue.
  - Closed
    CPP has been closed.
- **Active**
  Indicates, if CPP is active or inactive.
- **Inherit to submandators**
  Indicates, if case group of CPP inherits to submandators. If this is the case, submandators of the listed mandator are able to view this CPP and create CPPs for its case group.
- **Created by**
  User that created the CPP.

- **Created on**
  System timestamp of date the CPP was created.
- **Created by**
  Last user that has edited the CPP.
- **Created on**
  System timestamp of date the CPP was last edited.
- **Associated cases**
  Amount of cases that are associated with the CPP.
- **Closed as fraudulent**
  Amount of cases that are associated with the CPP and were closed with fraud status "fraudulent" .
- **Closed as genuine**
  Amount of cases that are associated with the CPP and were closed with fraud status "genuine" .
- **False alarms**
  Rate, that indicates how many associated cases have been closed with fraud status "genuine" in proportion to cases the CPP is assigned to.
- **Sum of evaluation attribute**
  In the case group definition of the CPP is an attribute specified as evaluation attribute. This value refers to the sum of all values of this attribute that occurred in associated cases that were closed as "fraudulent".

Remarks

- Additional columns are shown, if CPPs contain values for reporting attributes.
- Queries can be executed from within the table by clicking respective index attributes.
- CPPs must be active to be used.
- Notice that you can sort the investigation table by clicking on column headers. To sort for more than one column, simply click the columns in sequence (the former "inner" sorting will remain). Sorting preferences are stored with your user's account.

back to top

5.2.1 CPP

The configuration of CPPs

- **Active**
  A CPP can be active or inactive. Inactive CPPs cannot be assigned to cases.
- **Name**
  The name of the CPP.
- **Case group**
  The case group the CPP belongs to. The mandator belonging of the CPP is determined by the case group. Please note that the case group cannot be changed afterwards.
- **Status**
  The status of the CPP
- **Comment**
  A comment which is added to the CPP. Comments do not influence computation and are informational only.
- **Reporting attributes**
  CPPs can have multiple reporting attributes. You can choose the attributes you would like to assign and fill in values for each attribute in the appended form fields. Please note that you have to choose a case group first.

back to top

5.3 Queries

The table lists all queries that are defined for which you have access privileges.

Queries allow you to create any kind of extraction on transaction data stored in IBM Safer Payments. Queries defined here can be used in other parts of IBM Safer Payments, for instance to show past transaction records in case investigation.

back to top

5.3.1 Investigation Query

Investigation queries have the following settings:

- **Enabled**
  Lets you temporarily display or hide a query (notice that disabled queries are not shown on other pages where queries can be selected; they are still shown in the table below).
- **Mandator**

Each query belongs to one mandator. Once created, mandator ownership does not change.

- **Query type**
  IBM Safer Payments supports different types of queries:
  - **Ad hoc**
    This is the standard query type that returns all transaction records that satisfy the criteria defined. In spite of its name, the "ad hoc" query is always stored for future use when executed and must explicitly be deleted if not wanted anymore.
  - **Index**
    The "index" query type in addition takes the value of an indexed attribute as a parameter. It is typically used to define a transaction table for case investigation where the parameter would be the cardholder number or the merchant id. If you execute an "index" query from this page, IBM Safer Payments prompts you for the parameter value; if you execute such a query from another part of IBM Safer Payments, the value is automatically provided.
  - **Hyperlink**
    This type of query is similar to an "index" query. It is automatically executed when you click on an index value in a query result table.
- **Index**
  For "index" and "hyperlink" type queries, this references the index to be used.
- **Name**
  Name that will be shown with the query results.
- **Comment**
  Used to describe the query. The comment is displayed to users at various places and may thus contain further explanations.
- **Number of records**
  Limits the maximum number of records to be displayed to avoid excessive query computation length. Also notice that depending on your network infrastructure, the type of browser and computer used by the end user, the amounts of data generated by IBM Safer Payments could be overwhelming. If you have set IBM Safer Payments to use more than one thread to compute a query, the final number of records shown may actually be larger than this number.
- **Include DDC**
  If enabled, the query will also use DDC, not just MDC. This may severely impede query computational performance.
- **Hide summary statistics**
  The summary statistics for the query table can be disabled by checking this option.
- **Highlight CPP attributes**
  If enabled, attribute values that occur in CPPs will be highlighted in the query result table.

Query data selection

The data selection for queries allows for both choosing an interval and additional conditions. Refer to the section help pages for more information.

Select columns / column sequence

Allows to select which columns are to be displayed with the result of the query and how they should be arranged.

Extract template

Allows to define how the transaction data is put together in a string when a user is using the context menu function "extract data" on the query result table. Within the template fixed strings can be combined with variable attribute values.

example:

Arcot | {Amount} | {Merchant Name}

This template appends the transaction amount and the merchant name to a fixed string "Arcot", each separated by |.

A user could now select one or more rows in the query result table. By clicking on the context menu function "extract data" a new dialog is opened and displays a string in which the variable attributes of the template are filled with the respective transaction data. This string can now be copied to the clipboard [Ctrl]-[C].

Result set display

The results of a query when executed from this page are shown on a new page. If a query is used in another part of IBM Safer Payments ("embedded query"), the result table is shown as part of this page. In both cases, you may define which columns and in which sequence the columns are shown. The width of each column can be modified in the query result table.

back to top

5.3.1.1 Query Data Selection

Data selection lets you choose which mandator's data shall be included (if a choice from multiple mandators can be made) and lets you define interval and additional conditions. The interval can be provided as:

- Records absolute (URID from-to interval)
- Records relative (records from-to with respect to last inserted record)
- Server time absolute (from-to timestamp interval)
- Timestamp relative

Notice that the timestamps are taken from IBM Safer Payments server time at the time the record was created within IBM Safer

Payments (meta attribute "System time"), which is when the originating transaction was received (either as transaction message via the IBM Safer Payments message and command interface (MCI) or as file record processed via the IBM Safer Payments batch data interface (BDI)). If the record was later changed, for instance as merging target, this record timestamp value is never changed. Notice that these timestamps must thus not be the same as the time when the transaction actually was made (typically the "point of sales" type timestamp, a separate meta attribute "Timestamp" in IBM Safer Payments), since the transaction may have been received by IBM Safer Payments later (as in the case of batch data). If you instead require the "Timestamp" meta attribute to be used as a condition for your data selection, you must define it as a condition below. In this case, you should still consider using (applicable) time limits for the meta attribute "System time" as this allows IBM Safer Payments to sometimes significantly speed up the execution.

You may further restrict the records to be included using record specific attribute value conditions. Refer to their section help pages for more information.

back to top

5.3.1.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

5.3.2 Masterdata Query

Masterdata query definitions have the following settings:

- **Index attribute**
  Select the index dimension to which the masterdata element is associated.
- **Value**
  Enter the value of the index attribute for which the masterdata should be displayed. Formatting characters like digit group separator and decimal separators will be removed automatically

back to top

5.4 Group By Queries

The table in this section lists all group by queries that you can access.

Group by queries can be used to obtain statistical information about the indicators of a certain attribute. For each distinct value of the defined attribute that is stored in IBM Safer Payments' data caches, a statistical summary will be created. It displays the number and amount of genuine and fraudulent transactions as well as sums and averages. Additionally, it provides the possibility to analyse which accounts were encountered for which distinct value.

Group by queries can also be used to evaluate the performance of your current models. You can do so by using the option "Rule performance". This will collect the data of all rules that are defined with the setting "Performance report".

back to top

5.4.1 Group By Queries

On the upper part of the form, IBM Safer Payments provides a table of existing group by query results. You can access past query results by clicking on the respective row. The number of entries is limited and can configured.

Group by queries have the following settings:

- **Name**
  Name that will be shown with the query results.
- **Comment**
  Used to describe the group by query. The comment is displayed to users at various places and may thus contain further explanations.
- **Mandator**
  Each group by query belongs to one mandator. Once created, mandator ownership does not change.
- **Rule performance**
  If you would like to use this group by query to analyse the performance of your model revisions, you can do this by enabling this setting. Enabling rule performance analysis will override your group by attribute settings as these are mutually exclusive. To select a rule for performance analysis, you need to define the respective value on its definition form.
- **Group by attribute**
  To analyse the fraud distribution among the values (categories) of any attribute, you can select this attribute here.
- **Account analysis**
  Account analysis will prepare a breakdown which accounts had transactions containing the grouping value.
- **Include DDC**

If enabled, group by queries may access data that is available only on disk. Note that this might have severe impact on performance and that it is thus not recommended to activate this setting.

- **Show graph**
  If enabled, a graph will be appended to the result table. It is recommended not to use this setting for group by query results that have more than 10 groupings.

Timing analysis

The timing analysis setting will provide trending analysis for group by queries. You can select both the timestamp attribute and the resolution of the analysis:

- **Timestamp criterion**
  This attribute will be used for the trending analysis. Typically, the meta attribute timestamp is used here.
- **Resolution**
  Select which resolution will be used for the timing analysis. The resolution is calendar based, so if you select a daily resolution, all records that took place on the same day will be treated identically.

Group by query data selection

The data selection for queries allows for both choosing an interval and additional conditions. Refer to the section help pages for more information.

back to top

5.5 Common Point Queries

The table lists all common point queries that are defined for which you have access privileges.

Common point queries can be used to find equal values in transactions for certain index values.

Example:

There are three customer IDs 43211234, 12345678 and 87654321. The attribute CustomerID is indexed and has a sequence. The index has five entries for 43211234, three entries for 12345678 and two entries for 87654321.

CustomerID	Amount	MerchantID	Fraud
43211234	7.20	B3456	false
43211234	7.20	B3456	false
43211234	7.20	B3456	false
43211234	7.20	X1234	false
43211234	57.20	A7890	**true**
12345678	7.20	A4321	false
12345678	7.20	X1234	false
12345678	57.20	A2345	**true**
87654321	7.20	X1234	false
87654321	57.20	A6578	**true**

A common point query would find following common points for the common point attribute MerchantId:

MerchantID	Common points	Index values
X1234	3	3
B3456	3	1
A4321	1	1
A2345	1	1
A6578	1	1
A7890	1	1

X1234 was found in 3 transactions, it has 3 common points. All 3 common points had different customer ID, so this entry has 3 index values.

B3456 was also found in 3 transactions, it has 3 common points too. But all transaction had the same customer ID, so there is only 1 index value for B3456.

All three customers have reported fraud for transactions with different merchants, but they all have one transaction with merchant **X1234** as common point. It could be possible, that this merchant has leaked payment information and was the root for the following three fraudulent transactions.

back to top

6. Monitoring

This chapter covers the monitoring function of IBM Safer Payments.

back to top

6.1 Index Based Evaluations

The table lists all index based evaluations that are defined. Whether or not you can view or change index based evaluations is determined by the mandator roles you are granted.

Index based evaluations can be used to analyse historical data for each node of a given index and create alarms when certain conditions are fulfilled. Typically the index either represents an account or a customer, so index based evaluations can be used to implement KYC / AML procedures.

- An index based evaluation uses calendar profiles, events and masterdata to define alarm generation conditions.
- Calendar profiles are not used directly but via calendar computations defined within the index based evaluation.
- Index based evaluations also support multiple relations between two indexes e.g. between customers and accounts. In such a setup the index based evaluation runs through each primary index node (e.g customer) but checks the alarm generation conditions for each of their associated index nodes (e.g. accounts).
- Users who have the privilege can execute index based evaluations using a Job of type "Execute index based evaluation". A single job can execute a number of index based evaluations. Each evaluation will run isolated from the others so there will be no interaction between them.
    - Each index based evaluation must reference a case class of type "index based evaluation".
    - Created cases contain the result of each calendar computation defined in this index based evaluation at job execution time.

The following actions can be performed by using the context menu (right click)

- **Open definition**
  Open the definition of an index based evaluation.
- **Copy**
  Copy the definition of an index based evaluation to create a new index based evaluation with the same values.
- **Delete**
  Delete selected index based evaluation. This can also be applied to multiple rows.

back to top

6.1.1 Index Based Evaluation

This page describes the configuration of an index based evaluation.

- **Enabled**
  Only enabled index based evaluations will be executed by a job. Disabled evaluations will be skipped and can't be selected for new jobs.
- **Name**
  Name of the index based evaluation.

  Note: The name must be unique.
- **Comment**
  Comments are for documentational purposes only. It is advisable to comment the index based evaluation in a detailed way, so that the decision logic remains easy to understand.
- **Mandator**
  Each index based evaluation is assigned to a mandator.
- **Alarm type**
  Defines if the index based evaluation should create cases, notifications or both.
- **Case class**
  A case class of type "index based evaluation" must be selected to create cases.
- **Case score**
  A case score greater than or equal to 0 must be entered to create cases.
- **Notification**
  The notification to be sent. The available options depend on the selected mandator. Since index based evaluations are based on an index node, the message template can only reference attributes (using **{attribute name}**) and masterdata (using **[[masterdata attribute name]]**) belonging to the index or associated index (see section help for "Evaluate Multiple Values").
- **Index**
  An index needs to be chosen. When not evaluating multiple values this index decides which masterdata, events and calendar profiles are available.
- **Evaluate Multiple Values**
  Enabling this allows you to evaluate multiple values for each node of the primary index. A typical example would be to evaluate all customers and their associated accounts. Please see the section help for further information.
- **Calendar Computations**
  You may define one or more calendar computations to be used in the conditions below. Calendar computations become available in the conditions when all of their fields are filled with valid values.
- **Alarm Generation Conditions**

You may define one or more conditions that have to be met during execution of the index based evaluation to determine if an alarm should be created or not. You can use masterdata, events and calendar computations belonging either to the index or, when evaluating multiple values, the associated index.

back to top

6.1.2 Evaluate Multiple Values

Enabling this allows you to evaluate multiple values for each node of the primary index. A typical example would be to evaluate all customers and their associated accounts. To do this you would select an index on the customers as the primary index and use the additional options in this section to set up the associated data to evaluate.

- **Multiple value masterdata**
  Here you select a multiple value masterdata defined on the primary index which stores the values you want to evaluate for each node of the primary index. In our example you would select a masterdata containing all accounts for each customer.
- **Associated index**
  To find calendar profiles, events and masterdata associated with the values stored in the multiple value masterdata, you need to select an associated index to use. In our example this would be an index on the accounts. When selecting a multiple value masterdata defined with an associated index this field will be automatically filled out if it has been empty before.
- **Relationship Conditions**
  If the multiple value masterdata has relationship attributes you can define conditions on those to limit the associated index nodes that are evaluated. In our example you could use this to evaluate only accounts the customer is actually the owner of. When no relationship condition is defined, all associated index nodes belonging to the primary one will be evaluated.

Alarm Generation Conditions and Multiple Value Evaluation
When evaluating multiple values using the options above, alarm generation conditions will be evaluated for each associated index node belonging to a given primary index node. In our example all accounts for each customer would be evaluated as long as they fulfill the relationship conditions, if there are any. There are a few things to be noted:

- Calendar computations are calculated for each associated index node and then combined using the aggregation type defined. Those final values are then used to evaluate the alarm generation conditions.
- Masterdata and event values are evaluated for each associated index node. An alarm generation condition is fullfilled when at least one of those nodes matches it.

back to top

6.1.3 Calendar Computations

Calendar computations are intermediate values based on calendar profiles that can be used in the alarm generation conditions below. They become available in the conditions once all fields have been filled with valid values. Use the button [New calendar computation] to create new computations. Once created, each computation can be deleted using the [Delete] button in its top right corner.

back to top

6.1.4 Calendar Computation

Calendar computations are intermediate values calculated using a calendar profile. They can be used in the alarm generation conditions after all fields are filled with valid values. They offer the following settings.

- **Name**
  The name of the calendar computation.

  Note: The name must be unique within this index based evaluation.
- **Calendar profile**
  The calendar profile the computation uses. Only calendar profiles belonging to the index or the associated index (when evaluating multiple values) are available.
- **Period value**
  Calendar computations can either use the amount or the frequency stored in each period of the chosen calendar.
- **Computation type**
  Defines how the values of different calendar periods are combined.
- **Aggregation type**
  Only visible when evaluating multiple values. Defines how the computation results for each associated index node will be combined.
- **Past periods**
  The number of past periods this computation should use. Calendar periods are denoted by numbers: the value "0" corresponds to the current period, the value "1" to the first past period, and the value "(*n*-1)" corresponds to the oldest period (*n* is the "number of periods" as set in all calendar profiles' definitions). There are two ways you can define periods:
  - **n**
    If you just specify a single number, exactly this calendar period is used for the rule calculation.
  - **n~m**
    If you specify a number interval, all calendar periods of this interval are checked separately if they meet the rule. Notice that

this interval is inclusive, for instance, if you define "3~5", the calendar profiles' period settings is (are) "monthly" and the most current transaction was in mid-December, the periods checked in the rule would include the past months July, August, and September of this year.

back to top

6.2 Defined Risk Lists

The table lists all defined risk list definitions that are defined and for which you have access privileges.

Defined risk lists provide the possible to maintain attribute values (entities) that are either associated with high risk (block lists) or low risk (allow lists). While they are similar to model lists, there are a number of substantial differences:

- Most importantly, defined risk lists are not bound to model revisions, their entries exist independently. As a consequence, adding or deleting a defined risk list entry becomes effective immediately.
- Defined risk list entries are typically maintained by users that are involved in investigation tasks rather than by users that are involved in model generation.
- While lists within the model provide multiple operators such as *starts/ends with*, *contains*, or *close to*, defined risk list entries are only checked using the *equal to* operator. However, it is possible to define additional conditions which have to be satisified for each entry individually.
- While model lists create a new model attribute, defined risk lists assign the "output attribute value" to the "output attribute". The "output attribute" in this definition typically is another model "input" attribute.
- It is possible to easily add a start and an end date for each entry. With that it may be useful to have several entries with the same value. During computation all entries of a list are evaluated.
- Each entry on a define risk list can be disabled/deleted individually.
- Defined risk lists are optimized for large sets of entries.

Whether or not a user can view/change entries in defined risk lists is determined by the mandator roles he is granted.

The following actions can be performed by using the context menu (right click)

- **Open definition**
  Open the definition of a defined risk list at the below the table.
- **Copy**
  Copy the definition of a defined risk list to create a new, similar defined risk list.
- **Enable/Disable**
  Enable/Disable selected defined risk lists.
- **Delete**
  Delete selected defined risk lists. This can also be applied to multiple rows. Please note that you can disable defined risk lists instead of deleting them if you want to reuse them at a later time.

back to top

6.2.1 Defined Risk List Definition

The following settings are available to configure a defined risk list:

- **Name**
  Name that will be shown to access this defined risk list in the left navigation menu of the monitoring section.
- **Comment**
  Used to describe the defined risk list.
- **Priority**
  Using priorities it is possible to control the sequence of defined risk lists within a mandator. Defined risk lists within a mandator are evaluated in ascending order of the priorities. Possible priorities ranging from 1 to 10,000. Please note that first defined risk lists of mandators on higher levels within the mandator hierarchy are evaluated. Priorities only influence the sequence of defined risk lists within a mandator.
- **Mandator**
  Each defined risk list belongs to one mandator. Once created mandator ownership cannot be change.
- **Input attribute**
  Determines for which attribute the defined risk list is defined.
- **Output attribute**
  Defines which attribute is set by the defined risk list. This is typically an overwritable input attribute to the model revision.
- **Enable category selection**
  If the output attribute has categories, it is possible to choose the output attribute value per entry from the existing categories of the output attribute by checking this checkbox . In this case the output attribute value that is selected for this risk list, will be used as default output attribute value for new entries.
- **Output attribute value**
  Defines the value which is assigned to the output attribute if a transaction matches the entry of the defined risk list.
- **Expires**
  If enabled, each entry can be (optionally) configured together with an expiration date. Expiration dates can be configured for each

entry individually.

- **Default life time**
  If expiration dates are enabled, IBM Safer Payments provides the possibility to configure a default life time. If new entries are added to a list, the default expiration date will be the current date plus the configured default life time.
- **Starts at**
  If enabled, each entry can be (optionally) configured together with a starting date. Starting dates can be configured for each entry individually.
- **Explanation**
  The explanation text is displayed for users maintaining the defined risk list. You may thus use it to include specific instructions for the defined risk list.
- **Limit access**
  If enabled, the access to the entries of this risk list is limited to certain users.
- **Access authorized users**
  These are the users who may access the entries of the defined risk, if limited access is enabled. The entries of the risk list are not visible to users who are not selected here.

Enable for rule actions

Defined Risk lists can be enabled for rule actions. Using rule actions it is possible to automatically add entries to this defined risk list. Rule actions can be defined for any rule configured in the model. The following settings are available for the generation of entries by rule actions:

- **Comment**
  A comment which is added to each generated entry.
- **Label**
  A label which is added to each generated entry facilitates the identification for future search operations. (E.g. "Created by rule action")
- **Enabled**
  Defines whether entries created by rule actions are enabled or disabled. Disabled risk list entries are ignored during computation. You can disable defined risk list entries instead of deleting them, if you need to re-use the entry at a later time.
- **Replace existing entries**
  If this checkbox is enabled, existing entries with the same input attribute value as a newly generated entry will be deleted and replaced by the new entry. If disabled, it is possible to have multiple entries with the same value.
- **Filter criteria**
  Each entry of a defined risk list can be configured with additional conditions. Conditions that are defined in the section 'Filter Criteria' apply to generated entries. If additional conditions are defined only transactions that match the value for the input attribute and that satisfy all filter criteria are assigned with the configured output value.

Remark

In case "Expires" is enabled for this defined risk list, make sure that a proper default life time is defined since the default life time is applied as the expiration date for all entries created by rule actions.

back to top

6.3 Defined Risk List Entries

This table lists the defined risk list entries.

Defined risk lists provide the possibility to maintain lists of attribute values (entities) that are either associated with high risk (block lists) or low risk (allow lists). While they are similar to model lists, there are a number of substantial differences:

- Most importantly, defined risk lists are not bound to model revisions, their entries exist independently. As a consequence, adding or deleting a defined risk list entry becomes effective immediately.
- Defined risk list entries are typically maintained by users that are involved in investigation tasks rather than by users that are involved in model generation.
- While lists within the model provide multiple operators such as *starts/ends with*, *contains*, or *close to*, defined risk list entries are only checked using the *equal to* operator. However, it is possible to define additional conditions which have to be satisified for each entry individually.
- While model lists create a new model attribute, defined risk lists assign the "output attribute value" to the "output attribute". The "output attribute" in this definition typically is another model "input" attribute.
- It is possible to easily add a start and an end date for each entry. With that it may be useful to have several entries with the same value. During computation all entries of a list are evaluated.
- Each entry of a define risk list can be disabled/deleted individually.
- Defined risk lists are optimized for large sets of entries.

Defined risk lists offer several functions that are explained below:

- **Import entries from a file**
  Importing alert lists provides the possibility to quickly add entries to a defined risk list without manually creating them.
- **Add a new entry**
  You can add a new defined risk list entry manually.

- **Download table content as CSV-file**
  All entries that match the selection criteria of "Risk List Entries Selection" will be exported to a CSV-file. In case the selection contains sensitive data the CSV-File will be exported as part of an encrypted zip archive.
- **Delete all entries that are currently shown**
  All entries that match the selection criteria of "Risk List Entries Selection" will be deleted.
- **(De)activate all entries that are currently shown**
  Allentries that match the selection criteria of "Risk List Entries Selection" will be (de)activated. Entries that are already (de)activated will stay (de)activated.
- **Audit trail**
  All changes made to a defined risk lists are stored within the audit trail. Access to the audit trail is controlled by the respective mandator roles of users.

Whether or not a user can view/change entries in defined risk lists is determined by the mandator roles granted to users.

Remark:
The maximum number of rows which are displayed in the table is limited due to browser limitations. The maximum number of displayed rows can be configured on the page "Administration > System configuration > Defined Risk Lists".

back to top

6.3.1 Defined Risk List Entry Definition

The configuration of defined risk list entries

- **Value**
  The value of the defined risk list entry. Please note that you cannot change the value of an existing entry.
- **Output attribute value**
  If the output attribute of the defined risk list has categories and category selection was enabled in the defined risk list's definition, the output attribute value can be chosen from the output attribute's categories per entry. The preselected value is the default output attribute value of the defined risk list.
- **Comment**
  A comment which is added to the entry. Comments do not influence computation and are informational only.
- **Label**
  A label which is added to the entry. Usually a label is used to provide additional information. (E.g. "Import August 2013")
- **Enabled**
  A defined risk list entry can be enabled or disabled. Disabled risk lists entries are ignored during computation. You can disable defined risk list entries instead of deleting them, if you want to re-use the entry at a later time.
- **Starts at**
  In case the 'start at' option is enabled for the defined risk list, this field configures the starting date for the entry. The entry is valid once the starting date is reached (inclusive). In case this field is empty, no starting date is applied.
- **Expires at**
  In case the expiration of entries is enabled for the risk list, this field configures the expiration date for the entry. The entry is expired once the expired date is reached (inclusive). In case this field is empty, no expiration date is applied.
- **Filter criteria**
  Each entry of a defined risk list can be configured with additional conditions. If additional conditions are defined in the section 'Filter Criteria' only transactions that match the value for the input attribute and that satisfy all filter criteria are assigned with the configured output value.

back to top

6.3.2 Defined Risk List Import

Importing risk lists provides the possibility to quickly add entries to a defined risk list without manually creating them. The import file is expected to have one entry in every line and it has to be encoded in UTF-8. The input value has either to be followed by a line break / space character or the value has to be encapsulated in quotes. In case it is followed by a space character all characters between the space character and the line break are ignored. A header is not needed, in case a header exists it will be imported as a normal line.

- **Label**
  A label which is added to each entry facilitates the identification for future search operations. (E.g. "Import August 2013")
- **Comment**
  A Comment which is added to each entry.
- **Enabled**
  A defined risk list entry can be enabled or disabled. Disabled risk list entries are ignored during computation. You can disable defined risk list entries instead of deleting them, if you need to re-use the entry at a later time.
- **Starts at**
  In case the start of entries is enabled for the risk list, this sets the starting date of the entries. The entry is valid once the starting date is reached (inclusive).
- **Expires at**
  In case the expiration of entries is enabled for the risk list, this sets the expiration date of the entries.The entry is expired once the expiration date is reached (inclusive).
- **Replace existing entries**

If this option is activated, all existing entries with the same value will be removed and replaced by the new entry. If this option is not activated, another entry is added. In this case importing an already existing value to the risk list will lead to multiple entries with the same value.

A click on the save button opens a window where the file to import can be selected. Once the file is selected the import starts automatically. It is therefore important to fill in all fields before you select the file.
back to top

6.4 Compliance Lists

IBM Safer Payments supports all five elements of AML/CTF detection relevant to a payment processor:

1. Service risk classification
2. Customer profiling: hidden link analysis and usage classification
3. Geographical analyses
4. Transaction behavior
5. Monitoring of sanction/terrorist lists andpolitically exposed persons

While 1.-4. are implemented using standard IBM Safer Payments functionality within the model itself, IBM Safer Payments provides a specific functionality to monitor sanction and compliance lists. The details of this functionality are provided within this documentation.

Configuration process

To set up monitoring of sanction and compliance lists, the following steps have to be executed:

1. Activate monitoring: As a first step, monitoring has to be activated on the page *System Configuration*. IBM Safer Payments supports the OFAC sanction list, european sanction list, global watch list, United Nations sanction list, russian sanction list and the list of politically exposed persons. The mentioned lists can be activated individually. For more information refer to the online help of the respective section on the page *System Configuration*.
2. Specify raw data location: To be able to import activated sanction lists, the local storage of the raw data has to be specified on the page *Cluster*.
3. Configure user privileges: Privileges to view, add and edit compliance lists have to be granted to users. These settings are done on the page *Roles*. Refer to the respective online documentation for further information.
4. Import raw data: Sanction lists are imported to IBM Safer Payments during each startup. In addition, they can be (re-)loaded manually by clicking the respective button on this page.
5. Define compliance lists: After these preliminary steps, users with respective privileges are able to configure compliance lists both for processing real-time transaction and compliance ad hoc checks. To add a new compliance list, click on the button [New compliance list] to open a configuration form. Refer to the online documentation of this form for detailed information about the configuration of compliance lists.

Integration

Checks against imported sanction and compliance lists can be integrated into the real-time computation of incoming messages (using the BDI and/or MCI) or executed manually on demand by using the ad hoc check functionality. For both, automatic real-time checks and ad hoc checks, it is possible to create cases to further investigate alarms. In addition, notifications can be used to send messages to other systems or to create emails to be sent via SMTP in case matches are detected by IBM Safer Payments.

The workflow for automatic real-time checks is similar to defined risk lists. To integrate those checks into the real-time computation path of incoming messages the respective xml elements (or csv columns) - such as *name*, *passport number*, etc - have to be mapped to IBM Safer Payments input attributes. An additional attribute is needed to assign the computed score to. This attribute is then added to the incoming transaction and can be used within models to create alarms, cases and notifications. Compliance lists are defined for a specific mandator and can be restricted to types of messages by using computation conditions.

Compliance ad hoc checks are executed manually. This allows users to check people against sanction lists when this is needed. During ad hoc check, all activated algorithms are executed and a final score is displayed in case of hits. Refer to the online documentation of the page Compliance ad hoc checks for detailed information.

Compliance search also provides the possibility to manually search for sanction list entries. However, only the algorithm "starts-with" is executed and no final score is computed. Refer to the online documentation of the page Compliance search for detailed information.

Scoring process

When compliance checks are executed, names are used as primary keys to search for entries matching the name. To find entries on sanction lists matching the name sent within a message (or typed in manually for ad hoc checks) the following algorithms can be used. Each algorithm is configured with a score. In case an entry is found with more than one algorithm, the highest score of all fired algorithms is used:

- **Direct**
  This algorithm is always executed and cannot be deactivated. It creates a hit when the incoming name matches a name on the respective sanction list. The direct matching algorithm also uses some soft matching rules, e.g. double characters are reduced to single characters ("ss"="s"), matching similar characters ("z"="s", "w"="v", "c"="k"). See Remarks section for additional matching logic.
- **Starts with**
  As an example consider the entry *Chris Smith*. This entry will result in a "starts-with" hit for both the input *Christian Smith* and *Chris Smither*.

- **Metaphone**
  IBM Safer Payments provides the possibility to use a well known phonetic matching algorithm - *Double Metaphone*. Both names of list entries and incoming names are translated to double metaphone keys. For example, both *Christoph* and result in the double metaphone key *KRST*. Note that activating double metaphone algorithm can result in a huge amount of alarms. It is recommended to only use double metaphone only in combination with other fields and to only create cases if other fields result in matches too.
- **Levenshtein**
  The algorithm *Levenshtein* computes a similarity score based on the levenshtein distance. The computed similarity score ranges from 0 (no similarity) to 100 (direct match).

Remarks:

- The order of names (first/middle/last names) does not affect the result for any of the activated algorithms. *Hugo Christian Smith* and *Christian Hugo Smith* will result in a direct hit as well as *Smith Christian Hugo*. Names are splitted at white spaces and activated algorithms are executed on each part of the name separately. Assume that lists contain complete names, incoming names are considered a match when all parts of the incoming name result in a match for an entry. Consider the list entry *Hugo Christian Smith*. All subsets of these three names, such as *Hugo Smith* and *Christian Smith* will result in direct hits, whereas *Hugo Christian Alexander Smith* will not.
- Activating algorithms may influence the latency of the real-time process significantly. This depends mainly on the used algorithms and the size of the raw lists.
- In case an entry is found by multiple algorithms, the highest score is used for further computations.

After list entries were found using names as a primary keys, additional fields are checked if they are configured to be used. In case of matches for additional fields, such as passport number or date of birth, the configured score is added to the score which is computed during the check of the names. This results in a final score for each matching entry. The final score of the compliance score, which is assigned to the output attribute, is the maximum score of all matching entries. If a case is generated, all matching entries will be displayed.

Transliteration

All cyrillic letters are transliterated to latin letters using the ICAO (2013) standard. This is done for both, entries on sanction lists and incoming messages. This allows comparisons regardless whether cyrillic or latin letters are used.

Reload

During a reload of compliance lists, the raw data files (sanctions lists) for all configured compliance lists are imported and previously imported lists are deleted. Make sure that the respective files are available on all instances and file paths are configured correctly on the page 'Administration -> Cluster'. Note that a reload of compliance lists can take several minutes. Depending on your settings, transaction processing might be interrupted during the reload process of compliance lists. Only users with the respective privileges for compliance lists can trigger a reload of compliance lists. Reload of compliance lists is a global privilege that needs to be configured for each user account.

back to top

6.4.1 Compliance List

This page describes the configuration of compliance lists. For more information about compliance lists and their integration in the IBM Safer Payments computation process, open the general online help page Compliance.

- **Enabled**
  Compliance lists can be enabled or disabled. Disabled compliance lists are not executed for incoming messages. However, they can still be manually executed using ad hoc checks.
- **Name**
  Name of compliance list.
- **Comment**
  Comments are for documentational purposes only. It is advisable to comment the compliance list in a detailed way, so the decision logic remains easy to understand.
- **Priority**
  By using different priorities it is possible to control the execution sequence of compliance lists. Similar to rulesets and rules, compliance lists are computed in ascending order within each mandator.
- **Behaviour**
  Indicates whether it is online or offline. Offline compliance lists check the masterdata against the chosen sanction list and are run through the job scheduling function.
- **Entity type**
  Within sanction lists entities are separated in individual and legal entities.
- **Mandator**
  Each compliance list is assigned to a mandator. Similar to rules and profilings compliance lists are only executed for messages which satisfy the mandator conditions. In addition only attributes which are accessible by the chosen mandator can be used for compliance checks.
- **Threshold**
  An integer value is needed in case it is an offline compliance list. If the threshold is exceeded, a case will be created.
- **Case Class**
  A case class is needed if it is an offline compliance list to create cases when threshold is exceeded.
- **Score**
  A score for the case needs to be defined in case it is an offline compliance list.

- **Type**
  The type of list that will be used for sanction screening. Different types of lists can be configured under Administration -> System configuration.
- **Output attribute**
  Specify an attribute which is used as an output attribute for this compliance list. This attribute has to be defined in the model (as an input attribute) and has to be accessible by the chosen mandator.
- **Index**
  In case it is an offline compliance list, an index needs to be chosen to be able to get the associated masterdata.
- **Maximum number of hits**
  To limit the time which is needed by compliance checks, a maximum number of hits can be defined. IBM Safer Payments will stop the computation of compliance lists as soon as this number of hits is reached. Note that this can reduce the latency significantly but it can lead to situations where some potential hits are not recognized.
- **Additional matching algorithms**
  There are other algorithms available in addition to the default matching algorithm which is always executed and performs a string comparison between the incoming (or typed in) names and names from the sanction lists. These algorithms provide a "softer" matching for example to be able to detect entries with a slightly different spelling. Note that the activation of such algorithms has a big influence on the latency. Thus, contact the IBM Safer Payments support before activating any additional algorithms:
  - **Levenshtein:** A similarity score ranging from 0 (no similarity) to 100 (direct match) is computed using the levenshtein distance. When activating Levenshtein, a similarity threshold has to be configured. Entries with a similarity greater than the configured threshold are considered as hits.
  - **Metaphone:** Double metaphone keys are computed both for names included in sanction lists and for incoming names. These keys are used for comparison. Using (double) metaphone, it is possible to detect names used with different spellings such as *Chris* and *Kris*. Note that metaphone should only be used together with additional fields to avoid a large number of (false) alarms.
  - **Starts with:** Names are compared using the starts with operator. *Christian* and Chris are considered to be a match when "starts-with" is activated.
- **Ignore weak akas**
  If enabled, akas which are flagged as "weak" are ignored. This setting is only available for the OFAC sanction list.
- **Attributes**
  This section lets you define which attributes should be used as *name*, *street*, *city*, *country*, *passport number*, and *date of birth*. Note that the attributes together with their mappings have to be defined before they can be used within compliance checks. Values sent within these messages are used for checks against the respective fields of the sanction list. Note that only *name* is mandatory for compliance checks. All other fields are optional.
- **Score settings**
  Each activated algorithm is assigned with a score (an Integer number greater than 0). Usually the scores are descending with the "softness" (or matching accuracy) of the algorithm. In case an entry was hit by multiple algorithms, the maximum score is used. For each additional field (besides the field name) a score is defined. This score is added to the already computed score. In case an incoming message matches multiple entries on a compliance list, the highest score of all matching compliance list entries is assigned to the output attribute.
- **Computation conditions**
  By using computation conditions, it is possible to restrict the execution of this compliance list to specific messages satisfying the computation conditions. With that, different compliance checks can be used for different type of transactions.

back to top

6.4.2 Compliance Ad Hoc Check

You can execute ad hoc checks using compliance lists that are already defined and imported to IBM Safer Payments. Unlike the search functionality ad hoc checks evaluate the entered values in the same way as if they were sent within a message. That includes the processing of all enabled algorithms and the scoring logic. However, please note that the deactivation of compliance lists is ignored for ad hoc checks as well as computation conditions. To execute an ad hoc check at least a name is required. All other fields are optional.

- **Computed lists**
  Defines which lists should be computed. It is mandatory to select at least one list to execute ad hoc checks.
- **Name**
  You can enter the entity name to be checked against the selected lists.
- **All other fields**
  Except the name field all fields are optional. If a value is provided in an optional field it will be used for scoring and add to the overall score if there is a match. Empty fields and fields that are not enabled in the compliance lists will be ignored during computation.

back to top

6.4.3 Compliance Search

This page provides the possibility to search within the raw data of all loaded compliance lists. Unlike ad hoc checks the search functionality does not require a name. For example, it is possible to search for all entries from a specific country or with a specific passport number.

Compliance search always uses a starts with algorithm to find matching entries. In case you want to use other search algorithms such as Levenshtein please use the ad hoc check functionality.

- **Include list for compliance search**
  Defines which lists should be used to search for entries. Before using a list for compliance search the respective list has to be loaded to IBM Safer Payments.
- **Entity type**
  Defines whether individual or a legal entities should be listed.
- **All other fields**
  Using all other fields you can restrict the search results to entries that match the entered values. Leaving an entry field empty implies no restriction.

back to top

6.5 Merchant Monitoring Rules

The table lists all merchant monitoring rules that are defined and for which you have access privileges.

Merchant monitoring rules can be used to analyse merchants or acquirers by means of certain rule criteria. An example would be to find merchants which have been inactive (for which no transactions occured) during a certain number of consecutive calendar periods.

- A merchant monitoring rule in general uses calendar profiles or attributes, time ranges and thresholds to calculate whether one or more merchants met the rule.
- A rule uses one of several rule templates, each with its own input settings and its own calculation logic.
- For rules that use calendar profiles, the rule calculation logic takes amounts and / or frequencies of calendar profiles into account. For rules that use attributes, the rule calculation logic calculates along a sequence of records.
- Users who have the privilege can add merchant monitoring rules to "Administration/Jobs" when using "Generate report" as "Job type". With such a job csv exports can be created for the merchant monitoring rule. This export then contains the merchant monitoring report. Multiple rules can be added to a single job.

Whether or not a user can view/change entries in merchant monitoring rules is determined by the mandator roles he is granted.

The following actions can be performed by using the context menu (right click)

- **Open definition**
  Open the definition of a merchant monitoring rule at the below table.
- **Copy**
  Copy the definition of a merchant monitoring rule to create a new, similar merchant monitoring rule.
- **Delete**
  Delete selected merchant monitoring rules. This can also be applied to multiple rows.

back to top

6.5.1 Merchant Monitoring Rule

The following settings are available to configure a merchant monitoring rule:

- **Name**
  Name of the merchant monitoring rule. On csv export it will be shown as *"<type of the rule>: <name>"*
- **Comment**
  Used to describe the merchant monitoring rule.
- **Mandator**
  Each merchant monitoring rule belongs to one mandator. Once created mandator ownership cannot be change.
- **Type**

  Select between several types of rules. Each type provides its own input settings and follows its own rule calculation logic. In the rule types input settings you can select an acquirer index, time range(s) and the respective calendar profiles and thresholds or attributes. See the specific type's help for details.

  You can select between the following types:

  ```
  {0}
  ```

back to top

7. Model

This chapter covers the case model revision maintenance functions of IBM Safer Payments.

back to top

7.1 Model Revision Selection

This section explains how a model revision is selected.

back to top

7.1.1 Mandator Selection

The table below lists available mandators. Click on the respective row to access all model revisions of the respective mandator.

The details of IBM Safer Payments revision control is described on the help page of the model revision selection below.

back to top

7.1.2 Revision Control

In IBM Safer Payments, all settings dealing with how decisions are made are contained in a so-called "model". Therefore changes to the model can have significant effect on IBM Safer Payments operations. In order to control this, model changes are "managed" by a revision control system to ensure that changes can be audited and reverted. This section lets you perform actions on all model revisions of this mandator.

Filter settings

Let you select which model revisions are to be shown in the table using the drop-down list box above. Typically the "Active" setting is the most appropriate for everyday work as it hides all retired revisions.

Revision actions

To perform actions on a single model revision, click right on the respective row to open a context menu that offers (some of) the following actions:

- **View**
  Open revision in read-only mode.
- **Edit**
  Open revision for edit (causes the revision to be automatically reserved for the user until shared).
- **Share**
  Removes the reservation for this user.
- **Copy**
  Creates a copy of this revision as challenger.
- **Golive**
  Initiates the promotion of this revision into production.
- **Delete**
  Permanently deletes revision
- **Compare revisions**
  To compare two model revisions, hold the [Ctrl] key and select the revisions using left mouse clicks. Then click right to generate a report on the differences.

You may also open a revision by left-clicking on the respective row. If this revision was reserved for you, it will open in edit mode. Otherwise it will open in view mode.

Revision control

In most IBM Safer Payments applications, the model revision is constantly changed. Most changes are intended to counter emerging fraud patterns; others can be for various reasons, such as changes in the data structure or policy changes.

Nearly all such changes involve multiple elements, for instance a new counter is defined, an existing calendar profile is adapted, and a few new rules are inserted. If these changes were to come into effect immediately, there could be unwanted side effects during the time that the changes are made, and there would be no way to study the performance of the changed model beforehand. For instance, a model change to catch a new fraud pattern may in fact catch this fraud pattern well, but may generate an excessive number of false alerts, if it cannot be changed in advance.

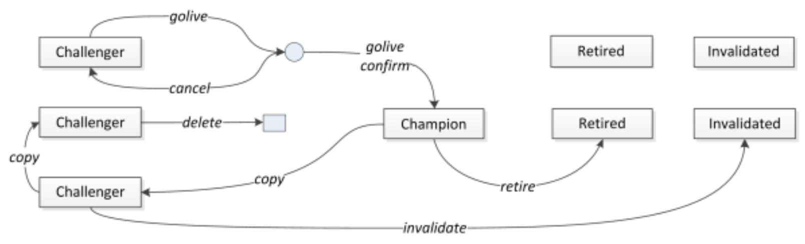
Therefore IBM Safer Payments allows users to "accumulate" the element changes within a so-called "challenger revision", and only accept them together into productive usage once all changes are complete and – typically – tested against past data to ensure that the challenger performs better than the champion.

Revision lifecycle

A new IBM Safer Payments installation is created with one (top) mandator and one champion revision of this mandator. With the champion revision, an initial model revision is defined. To perform statistical analyses on sample data and to check the performance of the model designed, IBM Safer Payments provides simulation capabilities directly in the model maintenance functionality.

Once the champion performs as desired, the so-called "golive" process is started. This process involves two steps. First IBM Safer Payments creates a "todo" list of every change this golive involves. These changes, and a detailed description of their consequences, are printed out as a report for the user. The user may now either accept and confirm the changes or cancel the golive process. If the changes are confirmed, the challenger becomes the champion. If the process is canceled, the challenger remains the challenger.

IBM Safer Payments always only executes the settings of the champion revision. If no champion revision exists, IBM Safer Payments ignores this mandator. If a challenger revision exists, the golive of a challenger promotes this challenger to be the new champion, and demotes the champion to be retired.



It is important to notice that if multiple challengers existed when one challenger was promoted to champion, the respective other challengers are invalidated. This is because once new challengers are derived from the now new champion; these *old* challengers could be inconsistent.

At any given time, there can be any number of challengers and any number of retired revisions per mandator, yet there can only be one champion revision.

Only challenger revisions can be edited and simulated. Challengers can be considered "candidates" that would be promoted to champions once they are considered to improve performance.

Challengers are created either automatically from scratch (with the addition of a new mandator), or as copies from champions or other challengers.

Because challengers are "work in progress", they can also be deleted. Any revision that was ever in production (i.e. a champion) may never be deleted, for auditing purposes.

The golive process in a cluster is described here.

Model inheritance

As pointed out in structural configuration, within a mandator structure, model revisions inherit all elements of their head mandators (i.e. all mandators on the path to the top mandator). Since each mandator in a mandator structure, however, has its own revision sequence, this has some implications:

- Mandator challengers only inherit elements of the champion revisions of their head mandators.
- In a challenger of a head mandator, an element can only be deleted if it is not used by any champion or challenger of its associated mandators.

Notice that the inheritable elements are listed in the table of the respective element type.

Info bar

Once a revision is opened, a vertical navigation menu on the left side of the page enables direct access to all elements of a model revision revision. While viewing or editing a revision, a horizontal info bar also appears directly under the main IBM Safer Payments navigation bar showing:

Edit|View [*RevisionNumber*] *RevisionName* of *MandatorName*

so that users are never in doubt on which revision they are working.

Revision numbers

Within each mandator, all revisions are automatically numbered. The numbers are generated sequentially for challengers at the time when they are generated as copy of a champion or another challenger to ensure that users have an easy way to tell the sequence of revisions generated.

Editing revisions

Each challenger can only be edited by one user at a time. For this, a user must explicitly set a challenger into "edit" mode. From then on, the revision remains locked to this user until the moment the user releases the lock ("share"). The locking is stored on disk, thus even a reboot of IBM Safer Payments will not release the lock. There is no explicit way to force unlock of a revision, however, the locked revision can always be copied as a new challenger.

Import/export

Because of the complex relationships between mandator revisions, IBM Safer Payments does not provide import/export features for revisions. To exchange an IBM Safer Payments configuration, you will need to copy the entire contents of the "cfg" directory.

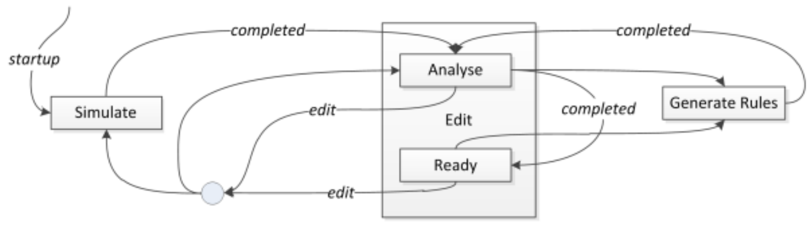
Auditing

Each revision control action can be logged in the IBM Safer Payments audit logs, where it is stored chronologically. In addition to this, the Revision section of the Model page for each revision shows when and by whom a revision:

- is currently being edited,
- was last changed,
- has been promoted to production ("golive"), and
- has been retired.

Challenger status

A challenger model revision is where all the analytical and model generation/adaptation takes place. Because of some major computational steps behind these steps, there are a number of status and status changes considered with each challenger.



On IBM Safer Payments startup, all challenger revisions are put into Simulate status (the same is true for a challenger revision that is generated as a copy of the champion revision). In Simulate status, the MDC values for all profiling output attributes are computed from the data of the input and profiling output MDC of the current champion (notice that simulation is limited to MDC available data to ensure that real-time performance is never compromised). This step is called "simulation" as it simulates a situation in which the respective attribute had always been there.

The typical usage for this is for a fraud analyst to devise for instance a new counter, and via simulation, be able to perform analyses on this counter output attribute value distribution, generate rules using this output attribute, and simulate performance of the model revision including this new profiling item.

During simulation, the model revision is not available for editing to avoid user error based on incomplete data. Because simulation is only performed in MDC, computation of simulation results is very fast.

If there are no profilings requiring simulations, the simulation step is skipped, and the revision status is set to Analyse. Here distributions of all attributes as well as rule performance statistics are generated. Because of the IBM Safer Payments incremental sample estimation technology, results are immediately available. Once the samples have reached 100%, the revision status is set to Ready. In both statuses Analyse and Ready, the decision logic revision can be edited. Any change to the revision that could have computational impact, however, restarts Analyses, or – if simulation is required – sets the status back to Simulate.

Once rule generation is started for a ruleset of a revision, the status is set to Generate Rules, and – with the exception of the generated rules themselves, no part of the decision logic can be edited until rule generation is completed.

Once a challenger is retired or promoted to champion, all simulation and analyses results are automatically discarded.

Status bar

Whenever you work with a decision logic revision, the small dark gray horizontal line under the main menu bar tabs is widened and provides quick information about the revision you are working with. These are the information parts for the various statuses:

- Performing simulation [*n*] *RevisionName* of *Mandator*
  Computation of simulated attributes in progress, view only.
- Edit (analyzing) [*n*] *RevisionName* of *Mandator*
  Computation of analyses in progress, editing possible.
- Edit [*n*] *RevisionName* of *Mandator*
  Standard edit mode.
- Generating rules [*n*] *RevisionName* of *Mandator*
  Rule generation in progress, view only.
- View [*n*] *RevisionName* of *Mandator*
  View only, no simulation, and no rule generation.

Notice that the status bar does not automatically update.

back to top

7.2 General Revision Settings

The upper section of the "general" page of a model revision provides the same actions for this revision in its toolbar as the context menu (right click) on the revision in the model revision table of the model tab page:

- **Save**
  Save name and comment settings.
- **Share**
  Unblock this model revision (so that other users can edit it).
- **Copy**
  Create a copy of this model revision (and reserve it for edit by me).
- **Golive**
  Initiate golive process by checking the necessary modifications and providing a report on all changes (this report must then be confirmed to initiate the actual golive process).
- **Model revision print view**
  Generate a model revision report in a new browser window. You can choose the scope of the report.
- **Delete**

Deletion of this model revision.

Notice that the actions available depend on the status of the model revision and your privileges. Refer to Revision Control for details.

The remainder of the section lets you name and comment this model revision, and view its status, and when and by whom it was set to a certain status.

xDC Statistics

The subsequent sections of the "general" page of a model revision provide an overview on the main memory and disk memory usages of the model revision.

The details on how IBM Safer Payments stores data is described in the Storage Architecture help page.

Audit Trail

The section Audit Trail shows an overview of the last 10 changes made to the currently viewed revision. Clicking the button in its toolbar shows a detailed audit trail.

back to top

7.2.1 Inherited Inputs

Input attributes shown in this section are inherited from mandators that are above this mandator within the mandator hierarchy. You may use these input attributes in the same way as the ones you define yourself (shown in a different section/table), but you may not change them.

back to top

7.2.2 Own Inputs

Input attributes shown in this section belong to this mandator and you may add, delete, and change these input attributes if you have the necessary privileges.

Notice that you must define mappings within this model revision to "connect" these input attributes to transaction messages coming into IBM Safer Payments for the attribute values to be populated.

back to top

7.2.3 Inherited Outputs

Output attributes shown in this section are inherited from mandators that are above this mandator within the mandator hierarchy. You may use these output attributes in the same way as the ones you define yourself (shown in a different section/table), but you may not change them.

back to top

7.2.4 Own Outputs

Output attributes shown in this section belong to this mandator and you may add, delete, and change these output attributes if you have the necessary privileges.

Notice that you must define mappings within this model revision to "connect" these output attributes to transaction messages generated by IBM Safer Payments for the attribute values to be included in message responses.

back to top

7.2.5 Input Attributes

Each attribute whose value is delivered into IBM Safer Payments that shall be used for real-time decision making or case investigation must be defined as an input attribute.

Notice that if you use a mandator structure, the rules of this decision logic may use all attributes defined in champion model revisions of mandators above in the structure.

Each input attribute is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain. Notice that the attribute names do not need to correspond to the variable names of data delivered to IBM Safer Payments; you define the relation between IBM Safer Payments attributes and variable names in "Mappings".
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic

remains easy to understand.

- **Storage type**
  Attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored of this attribute on disk.
- **Overwritable**
  If checked, rules may overwrite values of this input attribute.
- **Data type**
  IBM Safer Payments supports the data types:
  - **Boolean**
    Stores values of type "true/false" (stored using one Bit). If this attribute is not set its default value is "false". The default mapping for delivered data is "1" for "true" and "0" for "false". The mapping of the boolean values can be changed through preprocessings in "Mappings". Note: MDC or DDC capacity for boolean type attributes must be a multiple of 8.
  - **Numeric**
    Variable Byte length (0..8) signed numeric values with variable (0..6) decimals. Both settings determine the universe of the attribute that is calculated in the same form. If this attribute is not set its default value is "nil" for display and "0" for any computational use (as in a condition). You may also explicitly check this attribute for "nil" (empty) values in conditions.
  - **Text**
    Fixed length text values (configurable length). If this attribute is not set its default value is "nil" for display and "" (empty string) for any computational use (as in a condition). You may also explicitly check this attribute for "nil" (empty) values in conditions.
  - **Hexadecimal**
    Hexadecimal values (configurable length). Notice that the hexadecimal values can be up to twice as long as the Byte length defined. If this attribute is not set its default value is "00..." for any computational use (as in a condition) and for display.
  - **IPv4**
    IP address (e.g. 127.0.0.1) values (stored using 4 Bytes). Addresses can be delivered and are displayed as a text of four digit groups (0-255) separated by dots. Internally they are efficiently stored as binary information.
  - **Time interval**
    Two date and time values defining an inclusive interval (stored using 12 Bytes). Several types of date and time values are supported but both sides of the interval must use the same one:
    - Full timestamps
    - Time only
    - Day of the week only
    - Day of the week with time
    - Day of month and month only
    - Day of month and month with time
    - Day of month only
    - Day of month with time
  - **Timestamp**
    Timestamp (date and time) values (stored using 5 Bytes).
- **Formatted as**
  The formatting options are for display of values on the IBM Safer Payments pages (for examples in queries or case investigation). Choices differ by data type:
  - **Amount**
    Using digit group and decimal separators as defined for each user's preferences (e.g. "12,345.67") for numeric attributes only.
  - **Decimals**
    Using decimal separators as defined for each user's preferences (e.g. "12345.67") for numeric attributes only. This option does not use digit group separators.
  - **ID**
    Using digit group separators as defined for each user's preferences (e.g. "123,456,789") for numeric attributes only.
  - **PAN**
    Using dashed quadruple format typically used for primary account numbers as embossed on cards (e.g. "1234-1243-1243-1243") for numeric and text attributes only.
  - **No formatting**
    Shows data with no formatter applied.

  Notice that timestamp type attributes are always formatted according to user's preferences.
- **Encrypted**
  Encrypted attributes are stored on disk only in PCI DSS compliant format. User who do not have the privilege to view unmasked

data (global user privileges) will only see masked values of this attribute.

- **Purge entries**
  If checked, purging of outdated entries is enabled.
- **Length/decimals**
  Quantifies text and numerical data types:
  - **Numeric**
    Byte length of internal storage, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.
  - **Text**
    Byte length of internal storage, with ASCII coded characters, this is exactly the maximum number of characters that can fit into the attribute. Since IBM Safer Payments supports UTF-8 coding, non-ASCII characters may consume multiple bytes. For example, special characters in non-English European languages, such as ä, ü , ö, ß, ê, é, è etc typically require two bytes; all characters of Greek, Cyrillic, Coptic, Armenian, Hebrew, and Arabic require two bytes per character; and Chinese/Japanese /Korean Unified Ideographs require three bytes per character. You thus need to size the byte length of text attribute values according to the UTF-8 character encoding byte space requirements.
  - **Hexadecimal**
    Byte length of internal storage, thus any hexadecimal text value is twice the number of characters. For instance, the hexadecimal value "FF" would internaly be stored by IBM Safer Payments using one Byte.
- **Unit**
  Displayed with numeric values of this attribute. Typically used for currencies.
- **Meta attribute**
  IBM Safer Payments needs to know which of your (freely configurable) attributes represent certain fraud prevention standard attributes (aka "meta attributes") to render certain functions. Some meta attributes are mandatory (model revision will be refused for golive if missing), while others are optional. Because most meta attributes have pre-defined data types, the choice of the meta attribute depends on the type:
  - **Boolean meta attributes**
    - **Fraud (mandatory)**
      Used to mark transaction records as fraudulent. The value of 0 is considered "not fraud", while any other value can represent fraud or a fraud type. This attribute is read both for analyses and rule generation, and can also be set by the investigation function.
    - **URID computation complete (mandatory for Serialize Computation/Access protection)**
      The URID computation complete attribute is used internally by IBM Safer Payments when processing mergings, to know whether it is safe to use a transaction as a merging target.
  - **Numeric meta attributes**
    - **Account**
      Used to identify payment entities, such as cardholders or account owners. Typically this attribute is the PAN or the account number. This meta attribute can either be of numeric or text type.
    - **Amount (mandatory)**
      Used to identify the monetary value of a transaction record.
    - **Fraud (mandatory)**
      Used to mark transaction records as fraudulent. Using a numeric attribute as this meta attribute, two general configuration options are available. If no categories are defined for the attribute, the value of 0 is considered "not fraud", while any other value can represent fraud or a fraud type. As soon as one category is defined for the attribute, all categories that have been defined to be a fraud category are considered "fraud". In this case all other values are considered "not fraud". This attribute is read both for analyses and rule generation, and can also be set by the investigation function.
    - **Message type ID (mandatory)**
      Used to differentiate different types of transaction records and messages. Used for instance in "Mapping" to identify transaction message types.
    - **Primary instance ID**
      Used in an IBM Safer Payments cluster to identify the IBM Safer Payments instance that sets the primary URID (another meta attribute). Notice that transaction records may have different URIDs in each IBM Safer Payments instance. In order to uniquely identify each record, the primary URID in combination with the primary instance Id are used. Notice that the length of this meta attribute is fixed at 1 (ID values from 1 to 127).
    - **Primary URID**
      URID set by the IBM Safer Payments instance that had processed this record as primary instance. Notice that the length of this meta attribute is fixed at 8.
  - **Hexadecimal meta attributes**
    - **Account**
      Used to identify payment entities, such as cardholders or account owners. Typically this attribute is the PAN or the account number. This meta attribute can either be of numeric or text type.
  - **Text meta attributes**
    - **Account**
      Used to identify payment entities, such as cardholders or account owners. Typically this attribute is the PAN or the account number. This meta attribute can either be of numeric or text type.
    - **Email**
      If you use email/text notifications, this meta attribute indicates the attribute carrying the recipient's email address.
  - **Timestamp meta attributes**
    - **Timestamp (mandatory)**

This is the (IBM Safer Payments external) timestamp of the transaction. Typically this timestamp denotes the actual sale date and time. This attribute is used by IBM Safer Payments to understand the sequence of transactions irrespective of when they actually arrived in IBM Safer Payments.

    - **System time (mandatory)**
      This stores the IBM Safer Payments system timestamp that is automatically generated by IBM Safer Payments. This attribute is used by IBM Safer Payments to understand exactly when the transaction was processed.

- **Discard Future Timestamp**
  If checked, an attribute of type "Timestamp" will be discarded, if the timestamp is further in the future than the current timestamp plus the defined offset. The offset's time format is a combination of days and hours in which an upcoming timestamp is still regarded as valid.

  Notice that if you use a mandator structure, mandatory meta attributes must be defined in the top mandator.

Remarks

- If you enlarge the size of an MDC (or enable MDC) in a challenger, during golive, the "missing" data will be loaded from DDC to MDC.
- If you change the length or the number of decimals of an attribute **all stored information will be lost** once you start a golive.

back to top

7.2.6 Messages

Transaction data enters and leaves IBM Safer Payments as "messages":

- online transaction messages (those that require an immediate response such as authorization requests) use IBM Safer Payments's message control interface (MCI), while
- offline transaction messages use IBM Safer Payments's batch data interface (BDI).

More information on these interfaces can be found at the Interfaces Overview.

In a typical IBM Safer Payments application, multiple data sources (and drains) exist that all send transaction message requests to IBM Safer Payments. Since these messages typically stem from different source systems, they typically contain different data fields that require mapping of message variables to IBM Safer Payments input and output attributes. They sometimes even contain different data formats and hence require (possibly different) preprocessing.

Messages and mappings

IBM Safer Payments comprises full management capabilities for messages and mappings. Because the message definitions themselves are model revision independent, messages are defined on a mandator basis within IBM Safer Payments administration. Messages are inherited downwards within the mandator hierarchy. Based on the own and inherited messages defined, within each model revision, the mapping of message variables to IBM Safer Payments attributes and any pre-/post-processing is defined.

The various data source messages are identified by a MessageTypeId (aka MTID), which is a (mandatory) IBM Safer Payments meta attribute of numeric data type. Typical message types in an IBM Safer Payments application could include: authorization requests, masterdata delivery transactions, posted transaction notifications, fraud alerts, chargeback notifications etc. Each different message can have its own variable-attribute mappings and its own pre-/post-processing settings.

back to top

7.2.7 Mappings

In a typical IBM Safer Payments application, multiple data sources (and drains) exist. Frequently the IBM Safer Payments input and output attributes must be mapped (and sometimes pre-/post-processed) to the variables of these data sources. IBM Safer Payments thus allows for the definition of any number of messages from messages page on administration tab. For details refer to IBM Safer Payments Messages Online Help.

For each message and each input/output attribute, you can define which variable name and pre-/post-processing shall be used with the respective message. The respective table entry consists for XML/CSV/nested XML messages of *alias [preprocessing(preprocessingParameters)]* and for FCD messages of *start: length [preprocessing(preprocessingParameters)]*. For nested XML messages, the alias is represented by the sequence of elements that lead to the value starting from the root element. The different elements within the sequence are separated by a forward slash character (root_element/child_element/sub_child). If you want to map attributes, you would need to continue the sequence until the attribute (root_element/child_element/sub_child/attribute_key). In case there are array elements, they will be numbered starting from the 2nd element (array_element/item_2).

The same applies to JSON, except that arrays are represented as [x] where x is the position of the element in the array. (Example: transaction/order/items[0]/categories[0]/[0] for {"transaction":{"order":{"items":[{"categories":[["Books"]]}]}}})

In the table below, each row represents one IBM Safer Payments attribute.

Notes:

- Mappings of inherited attributes, recognizable through an entry in the Inherited from column, are not editable here.
- You can filter the attributes shown in the table according to their origin by selecting the respective option in the toolbar.

back to top

7.2.7.1 Mapping

The following settings are available for mappings:

- The mandatory "alias" identifies the name of the message variable that shall be mapped to the row's IBM Safer Payments attribute for XML online and CSV batch data deliveries.
- The mandatory "start" and "length" values let you specify the start position and length of this variable value of each data row for FCD offline data deliveries. Notice that the first character in an FCD row corresponds to start position of '1'.
- The "preprocessing" lets you specify a method of processing for this variable before it is mapped to the respective IBM Safer Payments attribute:
  - Any data delivery pre-processings
    - auto decimal
      Any message value delivered with no decimal period gets a decimal period inserted before *n*-th character from the right.
    - replace basic HTML entities
      Replaces &lt; &gt; &apos; &quot; &amp; with < > ' " &. Other HTML entities will not be changed.
    - counterfeit notes
      Computes a score to be used in conditions based on a list of denominations to be filtered, a specified score increment and a specified factor.
    - IP to hex
      Converts dotted-decimal IP address notation (127.0.0.1) to hexadecimal format (7F000001).
    - replace
      Defines *n* pairs of text values where the second text value replaces the first text value.
    - replace substring
      Each substring *texta* is replaced with textb in the value.
    - substring
      Cuts out the substring at a certain position with a certain length.
    - timestamp
      Accepts timestamp values of other formats than the standard IBM Safer Payments ISO format "YYYY-MM-DD hh:mm:ss".
  - Additional pre-processings for XML online and CSV offline data deliveries:
    - concatenate
      Appends the values of the listed variables(s) to this attribute.
    - crc32 hash
      Concatenates the values of the listed variables(s) and computes a crc32 checksum over the result.
    - evaluate travel periods
      Allows to perform a simple allowlist check for location and dates.
    - convert currency
      Converts the value using the integer variable value (alias) and converts it according to the integer *rate* and *decimals*.
    - take if empty
      If the alias defined is either not delivered with the transaction message or if it is delivered empty, a specific value is used instead.
  - Additional pre-processings for FCD offline data deliveries:
    - append constant right
      Appends a constant to the right of this attribute.
    - concatenate
      Appends parts of the FCD message together if multiple start length pairs are specified.

  Pre-processings can be used with any attribute type as they operate on string level only. They are applied to both transactions from batch files and from the message command interface.

back to top

7.3 Modelling

The process of creating model revisions assessing the likelihood of a transaction being fraudulent is referred to as "modelling". This section thus comprises all tools needed for this task:

- **Test**
  Testing allows to create "real-world like" transactions in a spreadsheet type metaphor, and have IBM Safer Payments compute its intermediate and output attributes from this. It is thus a quick and easy tool to verify behavior and to perform "what-if" type analyses.
- **Simulation**
  Simulation allows for the computation of new and modified model elements as if they had been in place with past time periods. In IBM Safer Payments, simulation is complemented with a large number of analytical tools to let you get a good understanding on how the model performs with the data.
- **Analysis**
  Statistical evaluation of original transaction data, derived attributes, and rule performance.
- **Rule generation**

Automatic and assisted rule generation let you generate complete models from scratch or adapt existing models to emerging fraud patterns.

These functions are directly available from the navigation menu left underneath the "Modelling" entry. The modelling page itself contains general configuration items that are used in all modelling functions.

The attribute modelling section allows for each attribute to define how it shall be used in:

- Test,
- Simulation,
- Analysis, and
- Rule generation.

To allow for attributes to be easily found even in models with multiple hundreds of attributes, the toolbar contains filters:

- Show/hide inherited attributes (toolbutton)
- Filter for certain attributes (drop list selection)

Notice that the attribute drop list is divided into two sections "origin" and "message attributes". If you check any of the "origin" entries, all attributes of this origin are shown in the table below. By selecting any of the (mandator defined) messages, all attributes that are mapped (with this model revision) in the respective message are also shown in the table below. If you select inputs as well as outputs in the origin section, the message section will be hidden, since you implicitly already selected all messages.

To change the modelling settings for test, analysis, and rule generation in details, simply click on the respective attribute row.

Remarks

Notice that you may highlight multiple rows in the table and open a context menu by a right mouse click. The context menu lets you enable or disable the selected attribute for any of the modelling uses. If no modelling selections for an attribute had never been made for this attribute, IBM Safer Payments will make an educated guess on the settings (that you can always change later); if previously modelling selections were made, they are remembered.

back to top

7.3.1 Settings

This section covers modelling settings.

back to top

7.3.1.1 Modelling

Each modelling setting defines if and how both own and inherited attributes are to be used in attribute analyses and rule generation (these settings do not influence how IBM Safer Payments makes decisions).

The choice of analysis model usages depends on the attribute's data type and on the nature of the attribute itself:

- **Categoric**
  This is best suited for attributes that have values on a nominal scale. In other words, the values just label concepts, and the order and absolute values have no meaning outside this. Typical examples include: merchant category codes, point of sale entry mode codes, country codes etc. The "max computed indicators" field lets you specify the maximum number of categories to be considered (to avoid overly lengthy computation times).
- **Integer**
  This is typically used for attributes that represent a counter where the number of events to be considered differently is not significantly higher than a few dozen (in which case the interval model usage might be better suited). The "max computed indicators" field lets you specify the maximum number of values -- which then is also the highest value minus one -- to be considered (to avoid overly lengthy computation times).
- **Intervals**
  This is best suited for attributes with continuous values with a near uniform distribution. For analyses and rule generation with this attribute, IBM Safer Payments creates a number of equally sized intervals. The value range in which the intervals are generated, and the size of the intervals, are specified by the from/step/to settings.
- **Logarithmic**
  Typically for all amount type numeric attributes. For analyses, this model usage creates logarithmic intervals using predefined 16 intervals of non-equal size.
- **Quantiles**
  This is similar to the logarithmic model usage of analysis. However, while a pre-defined log scale is best for the kind of exploration needed for analysis, for rule generation, the model usage "quantiles" is more appropriate as it creates intervals so that the total value of fraudulent transactions is roughly the same for all intervals. Notice that this model usage cannot utilise multiple CPU cores and may thus be significantly slower compared to other model usages. If you experience substantial slowdown, you may consider using custom intervals instead.
- **Custom Categories**
  This model usage lets you pre-define which categories are to be used. If you select this option, a section opens that lets you enter the values. This section also contains a more detailed online help function.
- **Custom Intervals**

This model usage lets you pre-define which intervals are to be used. If you select this option, a section opens that lets you enter the values. This section also contains a more detailed online help function.

The setting "max selected indicators" lets you define how many (single) indicators IBM Safer Payments would list in a proposed rule condition.

back to top

7.3.2 Test

Each model revision contains a "sandbox" type playground to test the revision's behaviour against data. You may (manually) enter any number of test records, name them, and provide them with any values for the attributes involved in decision making. Clicking the [Compute] button causes IBM Safer Payments to compute all non-entered values that are outputs of profiling or rules.

Test data records are stored permanently with the respective model revision (until they are manually deleted from the test page). The records are therefore copied together with a model revision when it is generated as a copy of another revision.

Records are strictly sorted by their "system time" meta attribute values (therefore no two timestamp values may be the same). This allows for profiling methods that consider time-sequences (calendar profiles, counters, events) to be computed from the test data records. Test never considers production data from the MDC/DDC for any of its computations.

Usage

- To create a new data record, click [New record] above.
- To enter values, double-click on the respective field.
- The pen icon indicates an entered value.
- To erase a value, double-click on the field and just delete the value.
- Click [Compute] to calculate all records.
- Unset and uncalculated values are set to their default values, which are indicated by round brackets.

Mandators

The decision models of all mandators are taken into account when computing the records. A green square 🟩 is shown in the column header of each test record that satisfies the mandator conditions for the mandator this one model revision revision belongs to.

Force profiling outputs

If you like to test the decision rule's computation to certain attribute value conditions that involve complex profiling outputs to be computed, you do not need to create all the records that result in specific profiling output attribute values. You rather may "force" them to a value by simply entering a value in the respective attribute's field. Notice that forced attributes have the same pen icon as entered input attribute values.

Remarks

- As soon as more than one test record is defined the meta attribute "system time" must be filled for every test record. All those timestamp values have to differ by at least one second. This is because the test function must know the sequence in which test records are to be considered. For a single test record this is not necessary.
- All entered values must be formatted using the same rules as used for transaction messages and batch file data. For instance, timestamp values must be entered as "YYYY-MM-DD hh:mm:ss", and numbers using the period as decimal character and no digit grouping symbols or dimension characters (such as currency symbols) may be used. Boolean values must be entered as "1" for 'true' and "0" for 'false'. Hexadecimal values can be entered using any combination of both lower and upper caps letters, e.g. "F0C4", "f0C4", and "f0c4".
- Computation considers the mandator hierarchy and mandator conditions:
    - A test record is only computed if the mandator conditions are met
    - If there are head mandators, their (champion) model revisions are computed before this (challenger) revision
- Computed values are stored as well. To re-compute, for instance after a rule change, click the [Compute] button on the test page. To remove all computed values from the table, click the [Reset computed values] button.

back to top

7.3.3 Simulation

Simulation allows for the computation of new and modified model elements as if they had been in place with past time periods. In IBM Safer Payments, simulation is complemented with a large number of analytical tools to let you get a good understanding on how the model performs with the data.

In a nutshell, simulation computes attributes and rules that have not been present in the champion model for past transaction data as if they had been present in the past. This includes both newly created and modified model elements. When simulation is first enabled, it analyzes which model revision elements must be computed to simulate the attributes and rules defined on the modelling page. Once simulation is computed, and a model revision element is changed, simulation is "stopped" automatically since results could be incorrect. When simulation is restarted after a model revision change, it detects which elements do require recomputation and only re-computes these.

Control

While which elements are to be simulated is defined on the modelling page, this page lets you control the actual simulation function using two toolbuttons:

- ▶ Start simulation
- ■ Stop simulation

The states of the icons represent the various states simulation can be in:

- Both icons disabled: simulation is locked since rule generation is under way.
- Only start icon enabled: simulation stopped and can be started.
- Only stop icon enabled: simulation is currently computed and can be stopped.
- Both icons enabled: simulation is computed. Clicking [Start simulation] refreshes the simulated data, clicking [Stop simulation] deletes all results and frees memory resources.

Notice that during simulation, a progress bar appears next to the toolbuttons. Rest the mouse pointer over the progress bar for details.

Simulation methods

IBM Safer Payments supports different simulation methods to complement different simulation objectives.

- By record
  This simulation method computes record by record so that any model revision element using any earlier record attribute value as input or in a condition, will be computed correctly. The disadvantage is that no parallel computing is used which results, compared to the computation method "in chunks", in a significantly lower simulation performance (speed in which results are available).
- In chunks
  This simulation method computes each record analogous to the method "by record" but subsequent records are simulated in parallel. The number of records which are computed in parallel is defined by the configurable number of simulation threads. In most real-world applications, because of the throughput of transaction messages, there should be so many "other" transaction messages between those of the same cardholder or merchant that feedback loops, for instance if a counter condition evaluates an output attribute or a rule overwrites an (overwritable) input attribute, should be fully represented. If this assumption is not valid for your application, you should use the simulation method "per record" instead.

Remarks

- IBM Safer Payments creates so-called "simulation MDC" as temporary storages for the simulated attribute's values that are released when simulation is halted.
- An attribute is simulated when:
  - it is enabled for simulation in modelling,
  - has an impact change and lies in the path of a model revision element that is simulated (e.g. a counter and its output attributes will be simulated, if the counter was changed with computation impact compared to the champion revision and the counter attribute is used in a condition of a rule that is simulated),
    note that the implicit simulation of elements in the path currently does not apply to PMML models so make sure to simulate a PMML model when needed by enabling one of its outputs for simulation,
  - it is used in a conclusion of a rule that is enabled for analysis.
- While simulation may take a while to be completed, during computation of the results, no partial results are available.

back to top

7.3.3.1 Model Simulation Data Selection

Data selection lets you choose which mandator's data shall be included (if a choice from multiple mandators can be made) and lets you define interval conditions. The interval can be provided as:

- Records absolute (URID from-to interval)
- Server time absolute (from-to timestamp interval)

Notice that the timestamps are taken from IBM Safer Payments server time at the time the record was created within IBM Safer Payments (meta attribute "System time"), which is when the originating transaction was received (either as transaction message via the IBM Safer Payments message and command interface (MCI) or as file record processed via the IBM Safer Payments batch data interface (BDI)). If the record was later changed, for instance as merging target, this record timestamp value is never changed. Notice that these timestamps must thus not be the same as the time when the transaction actually was made (typically the "point of sales" type timestamp, a separate meta attribute "Timestamp" in IBM Safer Payments), since the transaction may have been received by IBM Safer Payments later (as in the case of batch data).

back to top

7.3.3.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.3.3.3 Simulation Report

This report lists all actions that simulation will perform when you confirm this report. The color icons denote the type of report message:

- 🟥 This represents a situation in which there is potentially data missing. Only proceed if you know that this does not render your simulation faulty.
- 🟨 This represents a situation in which IBM Safer Payments feels that you made sub-optimal choices. Only proceed when you know that this is what you want.
- 🟩 Describes an element that IBM Safer Payments has determined that requires simulation and thus computational effort.
- No color icon represents an element that IBM Safer Payments has determined that does not require any computational effort since it is already available.

Notice that you may confirm simulation even if there are "red" marked entries since there are situations in which this makes sense.

back to top

7.3.3.4 Simulation Queries

The typical routine for adapting IBM Safer Payments to emerging fraud patterns is for the fraud analysts to experiment with modified rules or new/modified profilings. This activity is performed on a challenger model revision to the current champion model revision. Simulation is different from testing (on "test" page) as it involves (past) production data combined with results of profiling. This is why there is an extra function for "simulation".

Simulation provides a query-type functionality that is different from investigation queries in the sense that it combines all champion attributes with the (new) attributes of the respective challenger model revision.

Simulation is enabled and disabled from a toolbutton. If enabled, all applicable (new with this challenger) attributes are simulated. During simulation, you may not edit the model revision to avoid inconsistencies.

Notice that once simulation results are computed, they are kept until you disable simulation. Even if you have performed changes to the model revision in the meantime. To re-compute, disable and enable simulation.

Remarks

- Simulation computation works as a snapshot. When computation is started, the data record interval computed is from the current last record backwards. You may therefore re-execute the simulation after a while to ensure that the most current set of data is used for simulation.
- For champion model revisions, simulation is not available.
- Simulation requires IBM Safer Payments to create MDCs that exist while the model revision is in challenger status. IBM Safer Payments checks that sufficient main memory is available before it executes simulation and in addition monitors main memory availability through the respective SAI, however it should be kept in mind that improper use of simulation can exhaust the main memory made available by the hosting server hardware.

back to top

7.3.3.4.1 Model Revision Simulation Query

Data selection lets you choose which mandator's data shall be included (if a choice from multiple mandators can be made) and lets you define interval and additional conditions. The interval can be provided as:

- Records absolute (URID from-to interval)
- Server time absolute (from-to timestamp interval)

Notice that the timestamps are taken from IBM Safer Payments server time at the time the record was created within IBM Safer Payments (meta attribute "System time"), which is when the originating transaction was received (either as transaction message via the IBM Safer Payments message and command interface (MCI) or as file record processed via the IBM Safer Payments batch data interface (BDI)). If the record was later changed, for instance as merging target, this record timestamp value is never changed. Notice that these timestamps must thus not be the same as the time when the transaction actually was made (typically the "point of sales" type timestamp, a separate meta attribute "Timestamp" in IBM Safer Payments), since the transaction may have been received by IBM Safer Payments later (as in the case of batch data). If you instead require the "Timestamp" meta attribute to be used as a condition for your data selection, you must define it as a condition below. In this case, you should still consider using (applicable) time limits for the meta attribute "System time" as this allows IBM Safer Payments to sometimes significantly speed up the execution.

You may further restrict the records to be included using record specific attribute value conditions. Refer to their section help pages for more information.

back to top

7.3.3.4.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.3.4 Analyses

Using existing and simulated data of past transactions, IBM Safer Payments lets you define multiple analyses. Each analysis has multiple aspects that are computed and can be displayed by selecting the respective topic from the navigation menu left (below the entry "Analyses"). In this context, an "analysis" is a definition of a subset (or all) the data stored or simulated in IBM Safer Payments.

While often, defining just one analysis is all you need, sometimes it can be highly useful to have IBM Safer Payments compute multiple analyses at the same time:

- The definition of "training" and "verification" data selections is particularly useful when you create a fraud prevention model. A significantly lower performance of the model (or a part thereof) can for instance indicate overfitting.
- Sometimes it is useful to separate your analyses by criteria as "transaction type" (e.g. card not present vs card present, regions, merchant categories, card types, issuers). Since IBM Safer Payments computes all analyses in parallel, you simply can choose which analyses are to be compared in each topic.

Note that only those records will be considered for analyses where meta attribute timestamp is not empty.

back to top

7.3.4.1 ModelAnalysis.helpHeader.selection

ModelAnalysis.helpText.selection
back to top

7.3.4.2 Analysis

This form lets you define an analysis.

Notice that only enabled analyses are actually computed.

Analysis control

- Save modifications and start analysis
  Saves all modifications you made to this analysis and (re)starts computation. Previously computed results will be lost.
- Stop analysis
  If an analysis is in status "Computing" or "Optimizing" this button lets you stop the analysis without discarding the results. Notice that a stopped analysis cannot be continued.

Settings

- Rulesets
  Only selected rulesets will be included in this analyis. There are a number of reasons for not including all rulesets:
  - In certain applications, so-called "technical rules" are used. These are rules that do not detect fraud patterns but aggregate data for different purposes. There is thus no point in analyzing the correlation of such rules with the occurrence of fraud and hence the ruleset containing such rules typically are not included in modelling analyses.
  - In large models, the number of rules and rulesets can be so overwhelming that reducing the scope of a modelling analysis can be beneficial.

  Remarks:
  - You may analyze also inherited and final rulesets.
  - Only enabled rulesets are available for selection. Not enabled rulesets are not being simulated and can therefore not be analysed.
  - During next simulation, all rules of the enabled rulesets and the belonging conclusion attributes will be simulated. All other rules that change the simulated output attributes will be simulated as well. Additionally, other revision elements will be simulated, if they have computation impact changes compared to the champion and are used in conditions of simulated elements.
- Use simulation data selection
  If enabled, the analysis uses exactly the same data selection as defined for simulation. Notice that editing the data selection is not possible as long as this option is enabled.

back to top

7.3.4.3 Summary Statistics

Summary statistics provide a quick overview on the totals of the simulation/analysis data selection.

The following formulas are used:

- **Total:**
  (Genuine + Fraud)

- **Average:**

  (*Amount* / *Records*)

- **Fraud ratio:**

  The ratio is calculated in base points (BP):

  (((*Fraud* * 100) / *Genuine*) * 100)

The statistics include all records of the data selection of the analysis. The value of fraud is taken from the simulation result. Intercept, case queue, notification and reminder are computed as follows:

- Summary/Marked intercepted/Marked not-intercepted: The value for intercept is taken from production and is not altered by the current simulation.
- Rule fired: Only hits of rules that are enabled for analysis are counted. The hits are calculated based on the current simulation result.
- Intercepted by this model revision: The value of intercept is taken from the current simulation result of the meta attribute intercept. Note that all simulated rules have influence on this value, not only those which are enabled for analysis.
- Alarms/Notifications/Reminders generated: The computation includes only rules that are enabled for analysis. A record is counted, if there was at least one hit of a rule that creates a rule action or the meta attribute is set after the analysed rules have hit.
  Note that because only analysis enabled rules are taken into account, resetting a meta attribute by a higher prioritized rule that is not included in analysis, will not be reflected by the result.

back to top

7.3.4.4 Attribute Analyses

Attribute analyses can be executed for all attributes which are defined in this model revision.

Modelling parameters (and if attribute analyses is enabled or not) is configured on the "modelling" page of this model revision.

Notice that attribute analyses use a sampling technique in which extrapolated results are instantly shown even when only a fraction of the records are analyzed.

back to top

7.3.4.4.1 Attribute Analysis Chart

Attribute statistics are displayed graphically using bar/line charts. The options for these charts (display preferences, number of data points displayed, sorting, etc.) can be set from the tools in the section header (from left):

- The two show/hide bars/lines toolbuttons let you choose how the analysis is displayed.
- The "type of analysis" drop list lets you select what should be shown on the vertical axes:
  - Amount
    Shows total amount of all records for this attribute value (record amount defined by "amount" meta attribute).
  - Records
    Shows total number of records for this attribute value.
  - Ratio amount
    Shows ratio of total fraud record amounts divided by total amounts of all records for this attribute value (BP: "basis point", equal to 1/100th of a percent).
  - Ratio record
    Shows ratio of number of fraud records divided by number of all records for this attribute value (BP: "basis point", equal to 1/100th of a percent).
- Sorting criterion:
  - With "amount"/"record" type analyses of categorical model usage attributes, the displayed values can either be sorted by "fraud" or "genuine" amounts/records.
  - With analyses of categorical model usage attributes, the number of categories to be displayed can be selected here. Notice that the number of categories displayed can never be larger than the number of categories defined in the modelling parameters of the attribute itself.

With dots, bar and line charts, blue indicates fraud (right scale) while yellow indicates genuine (left scale). You may rest the mouse pointer over a dot/bar/line to display the exact values.

back to top

7.3.4.5 Rule Overlap

These statistics show the number of records from the selected analyses for which the conditions of multiple rules have applied, several rules have intervened, and alarms, notifications or reminders were generated by several rules. A strong overlap (high numbers) indicates difficulty in attributing a transaction interception to a single rule and thus should be avoided.

back to top

7.3.4.6 Rules Performances

The rule performance analysis plots key performance figures for each rule over time. The total time interval is defined by the analysis data selection entered.

You may select individually for which of the rules you like to see the analysis charted. Each ruleset defined therefore comes with its own checklist in which you can either select "all" rules or individual rules.

back to top

7.3.4.6.1 Rule Performance

The chart in this section plots the key performance figures you select on the check list to the right hand side over time. The time resolution can be changed by the drop list box in the toolbar, the time period can be changed by using the mouse to zoom into the period (highlight section with pointer or use scroll wheel).

Remarks

- On the left hand side, a scale is shown for any performance figured checked. The grid of the chart, however, is only shown for the first selected performance figure of the list.
- Hovering the mouse pointer over a data point gets you a readout of the exact value
- With weekly time resolution, the date value shown on the horizontal axis is the "wednesday date" of the respective week. The week ranges from Sunday 00:00:00 before this wednesday to Saturday 23:59:59 after this wednesday.

back to top

7.3.4.7 Rule Analysis

Single rule statistics benchmark the performance of each rule as if this rule would be the only one existing.

There are several statistics available for each rule of the selected rulesets. Note that you have to select a data range defined on the page 'Analysis'. All of the following statistics are computed based on the selected data range. IBM Safer Payments is using the mandatory meta attribute 'amount' to compute statistics including amounts.

- **Fraud transactions hit**
  Total number of fraudulent transactions hit by this rule.
- **Genuine transactions hit**
  Total number of genuine transactions hit by this rule.
- **Fraud amount hit**
  Total amount of fraudulent transactions hit by this rule.
- **Genuine amount hit**
  Total amount of genuine transactions hit by this rule.
- **Hit rate**
  Fraud amount hit by this rule divided by the total amount of fraudulent transactions within the selected data range.
- **False alarm ratio**
  Number of genuine transactions hit by this rule divided by the number of fraudulent transactions hit by this rule.
- **Saved amount per false alarm**
  Fraud amount hit by this rule divided by the number of genuine transactions hit by this rule.

back to top

7.3.4.8 Rule Optimization

Optimization analysis is based on single rule statistics. Hence optimization analysis does not start before statistics are completed. In optimization analysis, IBM Safer Payments simulates from the current set of rules, which rules when taken away would "optimize" the false alarm ratio to what extent.

IBM Safer Payments utilizes three different scenarios to compute the account based hit rate:

- **After hit**
  Any fraudulent transaction that occurs after a fraudulent transaction of the account was *directly hit* by an intercept is included in the calculation of the fraud loss prevented.
- **After intercept**
  Any fraudulent transaction that occurs after a transaction of the account was hit by an intercept is included in the calculation of the fraud loss prevented.
- **Any intercept**
  Purely account based calculation, the computed fraud losses sum up all account fraud if any transaction of the account was hit.

back to top

7.3.5 Automatic and Assisted Rule Generation

Standard model generation involves using statistical analysis to discover fraud patterns and a rule editor to create countermeasures. While this is well supported by IBM Safer Payments, there are two more levels of rule generation supported by IBM Safer Payments:

1. Manual model generation
2. Assisted model generation
3. Automatic model generation

For level 2. and 3., IBM Safer Payments employs a rule generating algorithm that creates rulesets condition by condition and rule by rule. The actual difference between level 2. and level 3. is that level 2. only suggests the next generation step and lets you modify IBM Safer Payments' proposal or not follow IBM Safer Payments' proposal at all, while level 3. assumes just an acceptance of each proposal and stops only once one of the stop criteria defined is reached. The remainder of this page provides some insight on how rule generation works, the actual sections on the pages that contain the respective rule generation functions have their own online help pages that contain usage details.

Indicators

Indicators are the smallest fragment of a condition and essentially are the possible values of an attribute (because a specific value of an attribute can be an "indicator" of fraud or not). Examples of indicators are:

- "CountryCode = FR" (categorical)
- "NumTrx1h = 0" (linear) and
- "Amount = [1220; 1480]"

IBM Safer Payments rule generation creates a list of indicators of each model attribute that can be viewed by clicking right on the respective row in the rule design table and selecting from the pop-up menu. This opens a dialog with a table where each row represents one of the indicators.

Conditions

A condition is an aggregate of one or more indicators. In simple terms, it combines an attribute with one or more values, or an open or closed interval (eg "MCC = 5141;5142", "MCC = 5142", "Amount > 1000", "Amount = [100; 600]", etc.).

Because a condition is the combination of all indicators selected, the condition can be seen and manipulated by selecting and deselecting indicators.

For each step in the rule generation process, the rule generation function assesses the individual quality of each indicator and derives a proposal for a condition. This process always starts by selecting one "seed" indicator for the attribute as the indicator with the highest quality.

Relaxation

The subsequent step is called "relaxation". Relaxation means to add indicators to the condition. How relaxation is performed depends on the type of attribute. With a categorical type attribute, any category is added that improves the quality of the condition. With linear and quantilize attributes, only relaxation is only performed with "neighboring" values (if for instance an intermediate linear attribute is used for model generation that counts shopping transactions within the past 48 hours, it would not make sense to add the indicator "Shopping=18" to "Shopping=12").

Condition selection

Rule generation proposes one of these actions:

- Commit condition
  In the rule design table, the respective condition row is highlighted in yellow and the [Commit condition] button is enabled. The IBM Safer Payments recommendation is to add this condition to the currently designed rule.
- Commit rule
  In the rule design table, no condition row is highlighted in yellow and the [Commit rule] button is enabled. The IBM Safer Payments recommendation is to add the rule as defined by the pushpinned conditions to the ruleset and start creating the next rule.
- No action
  Both the [Commit condition] and [Commit rule] buttons are disabled in the rule design section, and no rule is highlighted in the table. IBM Safer Payments recommendation is to stop rule generation based on the defined parameters.

Notice that automated rule generation can be enabled and disabled at any time. As long as automatic rule generation is disabled, you may create rules by selecting indicators for conditions, conditions for rules, and rules for the ruleset.

Remarks

- Rule generation only uses the per-transaction fraud figures. Depending on the specifics of the data, the per-account fraud figures may be quite different. However, since in general, rules that are good in finding per-transaction fraud are also good in finding per-account fraud, this should be no problem in practical applications. To determine the exact numbers for per-account fraud, use the account analysis function.
- Notice that all fraud figures shown in the generated rules table assume that each subsequent rule is used in addition to the one before. If a rule is used "alone", it can have significantly different performance. Use the ruleset optimization analysis to find an optimized sequence and subset of the generated rules. The rule generation function generates rules, the rule analyzer shows optimal subsets of the rules for a decision logic.
- Generated rules are not automatically transferred to the decision logic as a matter of precaution. You select the import function at the conclusion of the rule generation.

back to top

7.3.6 Rule Generation Settings

The IBM Safer Payments rule generation function employs a number of complex algorithms to propose and generate rule conditions and rules that detect fraud with high accuracy. Since each application has its unique fraud patterns and individual policy, the rule generation algorithms can be fine-tuned to fit the specific needs by the parameters set in this form. Refer to the Automatic and Assisted Rule Generation page for details on the approach behind this. Notice that the parameters that control rule generation that deliver the best results are highly dependent on your application, the structure of your data and fraud patterns. Please consult the IBM Safer Payments support for the best set of settings for your applications.

The settings on this page are:

Rule name / comment

These fields allow the entry of a text with variable elements that is used to name and comment the generated rules. Variable elements include:

- {n}, {nn}, {nnn}
  Number of rule in ruleset, {nn} fills it to two digits, {nnn} to three digits so they can be sorted by name.

For comments only, variable elements also include:

- {localtimestamp}
  ISO formatted timestamp of the rule generation in the time zone of the user.
- {revision}
  Number of revision.
- {timestamp}
  ISO formatted timestamp of the rule generation in in UTC.
- {user}
  Name of user.

Weighting (Default = 0.0, value range -0.999 to +10.0)

Weight determines how much IBM Safer Payments shall favor hit rate over false positives. The value of 0 is the default settings in which IBM Safer Payments tries to consider a good compromise between both a high hit rate and a low false alarm ratio. If you decrease this value below zero, rules will be preferred that have a low false alarm ratio. If you increase this value above zero, rules will be preferred that have a high hit rate (Thus a rules set will have less rules the higher this value is).

Min false positives (Default = 1.0, value range 0.0 to 10.0)

In rule generation, sometimes indicators, conditions, or rules exhibit a very low false alarm ratio. The reason for this is often overfitting because only a few transactions are concerned. As a result of this very low false positive rate, these elements would get a very high evaluation that would steer rule generation "too much" toward these elements. Therefore this setting allows the setting of an "indifferency" zone: any false positive value below this value is considered to be indifferent to this value by IBM Safer Payments.

Stop false positives (Default = 50.0, value range 1.0 to 500.0)

Automatic rule generation and rule generation proposals are stopped if the last rule generated has false positives equal or greater than this setting. This setting is mostly used as a stop criterion for automatic rule generation as the creation of a rule with a very high false alarm ratio usually is a signal that no good quality rules can be generated that improve the model performance.

Upper bound hit rate (Default = 50.0%, value range 1.0% to 100.0%)

Automatic rule generation and rule generation proposals are stopped if the total of rules generated hit this percentage value in hit rate. This setting is mostly used as stop a criterion for automatic rule generation.

Max number of conditions (Default = 6, value range 1 to 12)

Defining a max number of conditions that are generated for each rule avoids the generation of overly complex rules. However, notice that IBM Safer Payments rule generation may require that to get the false alarm ratio down more conditions must be allowed. If the number of conditions is too limited, IBM Safer Payments may not be able to generate rules with a sufficiently low false alarm ratio.

Max hit (Default = 50.0, value range 0.1 to 100.0)

In rule generation, sometimes indicators, conditions, or rules exhibit a very high hit rate. The reason for this is often overfitting because only a few transactions are concerned. As a result of this very high hit rate, these elements would get a very high evaluation that would steer rule generation "too much" toward these elements. Therefore this setting allows to set an "indifferency" zone: any hit rate value above this setting is considered to be indifferent to the value.

Max number of rules (Default = 50, value range 1 to 1000)

Number of rules to be generated as stop condition.

Min sample size (Default = 0.01%, value range 0.0% to 10.0%)

Ensures that only conditions are considered that cover at least this percentage of transactions of all training data transactions (genuine and fraudulent). Increase this value if you experience that rules are generated that fit only very few transactions and thus are prone to overfitting.

Min condition improvement (Default = 0.2, value range 0 to 10.0)

IBM Safer Payments uses a goal function for rule quality on a relative level. Adding a condition to a rule changes the quality value determined by this function. This setting defines the minimum improvement to the quality value a new condition must bring to a rule for it to be considered. Having a non-zero threshold ensures that conditions are only added to a rule if they provide significant improvement of its performance. This ensures that rules remain both easy readable by a human and avoids overfitting. Increasing this value results in fewer conditions per rule and potentially lower quality rules, decreasing this value results in more conditions and potentially overfitting. If no condition would improve the current rule by this threshold, no condition is selected.

Min population quantise (Default = 1000, value range 10 to 10,000,000)

If the model usage for a continuous variable (ratio measurement scale) is "quantilise" and a number of quantiles to be generated is defined (modelling), the number of quantiles will automatically be reduced if the defined number of transactions (genuine and fraudulent) is fewer than this setting. This ensures that no indicators are generated (and conditions constructed from them) that have so few records in them that overfitting is probable.

Min select hit rate (Default = 0.3%, value range 0.0% to 10.0%)

To avoid overfitting, rule generation will not recommend the selection, if the condition catches less than the defined part of the total fraud. If none of the model attributes contains indicators that can be used to design a condition that catches more than the defined part of the total fraud, rule generation proposes to end the construction of the rule. This criterion is hence crucial to define how fine vs. coarse the generated rules shall be. By decreasing this parameter, the rules generated will in general have a lower false alarm ratio but you will need more rules to catch the same total of fraud and thus may deliver better performance. Typically the rules will have more conditions. More rules with more conditions make a less easy to read rule base and bear the risk of overfitting. The optimum setting of this value depends on the structure of the fraud in your data.

Relax All Up Threshold (Default = 0.5, value range 0.0 to 10.0)

Some attributes represent indicators that are known to indicate a higher risk the higher their value is. Examples for this are counters that identify risky circumstances with the previous transactions. Because frequently, the transaction histories involving high scores of such indicators are rare even with large simulation data volumes, the rule generation function will create condition intervals with upper bounds for such attributes, even though an interval with no upper bound would be more appropriate. You may choose to perform this “relax all up” function automatically, using a lower value of this parameter. Rule generation compares the quality of the condition interval with an upper bound to one unbounded. If the unbounded does not deliver a quality that is worse more than the value of this parameter, the indicators are all relaxed up to an unbound interval. Notice that the unit of this parameter is the same as with the "min condition improvement" parameter explained above.

back to top

7.3.6.1 Rule Generation Data Selection

This section defines which data is to be used for rule generation. Notice that you may return to this page from rule generation to either check your settings or to change them. In the case that you change data selection settings, the selection is recomputed, which may take some time.

Rule generation scenario

This setting defines the general scenario for which rules are generated:

- **Ignore existing rules and ignore existing intercepts in records**
  Creates a fraud prevention model as if there were no records marked "intercepted" in the data and no rules were already defined. Choose this scenario if you are creating a completely new set of prevention rules ("from scratch").
- **Ignore existing intercepts but consider already defined rules of this model revision**
  Choose this scenario if there are already defined rules in this model revision that should be considered. This keeps IBM Safer Payments from creating them (or similar rules) again.
- **Consider already defined rules of this model revision and already existing records intercepts**
  In addition to records that are already hit with existing rules, this scenario also considers records that have a non-zero value of the "intercept" meta attribute. Choose this scenario if you are in daily operations with IBM Safer Payments and you want the rule generation to only suggest fraud countermeasures (rules) based on the evaluation of records that have not been intercepted in the past (with whatever model revision was in production then) and that are not hit by rules already defined (and enabled) in this model revision.

Verification

If enabled, IBM Safer Payments performs automatic verification of rule generation results. Enabling this checkbox opens a second data selection subsection for the verification data. Typically you would select an initial time/data period for training, and a subsequent time/data period for verification.

Verification results are displayed in green next to the training results (in black). Typically verification results are not as good as training results. This is because the fraud patterns in the verification data selection are somewhat different to the fraud patterns of the training data selection, and thus, the fraud countermeasures (rules) developed for the training data selection only perform to a fraction for the verification data. This effect is called "overfitting", meaning that the rules fit the training data better than the verification data. It is an inevitable part of any model generation. However it is important that the degree of overfitting is within limits because otherwise the models you create with training data will not be useful.

Which degree of overfitting is acceptable (i.e. must be lived with) strongly depends on your application. Notice that overfitting is strongly influenced by the rule generation settings and the profilings defined. Consult with the IBM Safer Payments support to identify the proper settings and overfitting criteria for your application.

Training/verification data selection

These subsections allow for the definition of exactly which data is to be used for training/verification. The data range settings let you select a from/to type range, and the attribute condition subsection lets you in addition to that also define per-record criteria.

Remarks

- Notice that the "performance" section (top table) of the rule generation page shows the totals resulting from this selection.
- More information on can be found on the Automatic and Assisted Rule Generation online help page.

back to top

7.3.6.2 Rule Generation

The upper section of the rule generation page shows the performance table, a collection of statistical data about the rule generation process. It is only displayed once all results are completed. As long as the results are still computed, a progress bar is shown instead of the results. You may also change the data selection and rule settings from here by clicking on the [Change data selection settings] and [Rulesettings] icon.

Notice that if verification is enabled, all respective values in the table are shown underneath the training data values in green.

Columns

1. Total
   Total amount (taken from the "amount" meta attribute) and number of all records that are fitting the training data selection criteria defined.
2. Fraud
   Total amount (taken from the "amount" meta attribute) and number of all records marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
3. Genuine
   Total amount (taken from the "amount" meta attribute) and number of all records *not* marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
4. Statistics
   Contains various derived performance indicators:
   - Hit rate
     Total fraud amount of this row's data selection divided by the total fraud amount in all records fitting training data selection criteria.
   - False positives
     Number of genuine records falsely hit divided by number of fraud records correctly hit in all records fitting training data selection criteria.
   - Saved amount per false alarm
     Monetary savings by fraud prevented for each false alarm endured (refer to Benchmarking Prevention Performance for details).
   - Intercept
     Number of records hit divided all records fitting training data selection criteria (displayed as "basis points", 100 BP equals 1%).

Rows

1. All
   All records that are fitting the training data selection criteria defined. Notice that the "hit rate" for this, by definition, is always 100%.
2. Marked intercepted in data
   All records for which the "intercept" meta attribute is non-zero (in other words, that IBM Safer Payments in the past had intercepted). Depending on the rule generation settings, these records are missing from row 4.
3. Intercepted by existing rules
   All records for which any rule in the current model revision (and any inherited rules) fires (in other words, records that IBM Safer Payments would mark intercepted with the current rules). Depending on the rule generation settings, these records are missing from row 4.
4. Used for rule generation
   All records that are used to generate the rules. These are either all records of row 1., minus the records of row 2., minus the records of row 2. and 3. (depending on the rule generation settings).
5. Intercepted by generated rules
   Performance of the rules already generated (these are the rules listed in the third section of this page). This row is only shown when rules are generated.

Remark

More information on can be found on the Automatic and Assisted Rule Generation online help page.
back to top

7.3.6.3 Rule Designer

The middle section of the rule generation page shows the rule design table, where the actual rule generation mostly takes place.

Notice that even if verification is enabled, no verification data is shown. This is because verification data is about verifying the rule design decisions, not influencing them.

Toolbar

- [Commit condition]
  If enabled, IBM Safer Payments proposes (yellow highlighting) a next condition for the current rule (under construction). Clicking this icon adds the condition to the rule.
- [Commit rule]
  If enabled, IBM Safer Payments proposes to create a rule out of all conditions that are pushpinned. Clicking this icon adds the rule to the set of newly generated rules.
- [Start fully automated rule generation]
  If enabled, IBM Safer Payments lets you activate the automated rule generation mode in which each IBM Safer Payments proposal is accepted. While the automated rule mode is running, the icon shows a small green dot. Clicking on it again, stops the automated rule generation mode. You may then either create rules manually or start the automated mode again.
- [Change rule generation settings]
  Allows you to change the rule generation settings during rule generation. Changing settings typically only results in minor re-computation efforts.
- [Reset condition]
  If you have created one or more conditions of the current rule, you can delete them by clicking this icon.
- [Hide/Show Attributes without selected indicators]
  Adjusts the view of attributes to only show attributes with selected indicators or show all.

Columns

1. Attributes
   Name of the model attribute. Add model attributes or remove model attributes by clicking the [Change attributes model usage] icon above. Left of the attribute name, the blue arrow icon indicates that this attribute is proposed by IBM Safer Payments to be used for the next condition (row is also highlighted in yellow); a pushpin icon indicates that this attribute has already been used as a condition attribute in the current rule.
2. Usage
   Attribute usage as defined in the modelling settings. Click the [Change attributes model usage] icon above to change.
3. Indicators
   Delivers information about the indicators for this attribute:
   - Selected
     Number of indicators currently selected for the best condition IBM Safer Payments proposes for this attribute.
   - Total
     Total number of indicators found and considered by IBM Safer Payments for this attribute.
   - Limit
     Maximum number of indicators considered by IBM Safer Payments for this attribute.
4. Condition
   Best condition IBM Safer Payments proposes for the respective attribute:
   - Op
     The condition operator.
   - Constant
     The condition constant.
5. Total
   Total amount (taken from the "amount" meta attribute) and number of all records that are fitting the training data selection criteria defined.
6. Fraud
   Total amount (taken from the "amount" meta attribute) and number of all records marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
7. Genuine
   Total amount (taken from the "amount" meta attribute) and number of all records *not* marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
8. Statistics
   Contains various derived performance indicators:
   - Hit rate
     Total fraud amount of this row's data selection divided by the total fraud amount in all records fitting training data selection criteria.
   - False positives
     Number of genuine records falsely hit divided by number of fraud records correctly hit in all records fitting training data selection criteria.
   - Saved amount per false alarm
     Monetary savings for fraud prevented for each false alarm endured (refer to Benchmarking Prevention Performance for details).
   - Intercept
     Number of records hit divided all records fitting training data selection criteria (displayed as "basis points", 100 BP equals 1%).

Rows

Each row represents one model attribute. The IBM Safer Payments rule generation algorithm creates at each step a possible condition for each attribute/row that it considers the best "it can make from this attribute". Which of the conditions/rows it considers as best next steps is highlighted in yellow.

You may now accept any of the conditions for the current rule by selecting a row and using [Commit as condition] from the context menu, you may accept the highlighted choice by clicking the [Select condition for *attributename*] toolbar icon, or you may modify any of the conditions by selecting [Explore all indicators] or [Explore performing indicators] from the row' pop-up menu.

Notice that when you change any indicators of a condition in the "indicator" dialog that opens from exploring, the effects become visible immediately in the right columns of the row to provide immediate feedback on the action.

Remark

More information on can be found on the Automatic and Assisted Rule Generation online help page.
back to top

7.3.6.4 Indicators

The table on this dialog shows the indicators for the respective attribute.

Notice that even if verification is enabled, no verification data is shown. This is because verification data is about verifying the rule design decisions, not influencing them.

Columns

1. Sel(ection)
   Checkbox determines if this indicator is part of the condition for this attribute. Notice that if you select/unselect any indicators, the changes to the performance of the entire condition are immediately visualised in the rule design table (from which the dialog was opened). Notice that for certain usage types, only adjacent indicators may be selected (to avoid fragmented conditions). Also notice that all changes made are immediately reflected in the rule design table, so all you need to do is close the dialog once you have made all desired changes.
2. Indicator
   Value of the indicator.
3. Total
   Total amount (taken from the "amount" meta attribute) and number of all records that are fitting the training data selection criteria defined.
4. Fraud
   Total amount (taken from the "amount" meta attribute) and number of all records marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
5. Genuine
   Total amount (taken from the "amount" meta attribute) and number of all records *not* marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
6. Statistics
   Contains various derived performance indicators:
   - Hit rate
     Total fraud amount of this row's data selection divided by the total fraud amount in all records fitting training data selection criteria.
   - False positives
     Number of genuine records falsely hit divided by number of fraud records correctly hit in all records fitting training data selection criteria.
   - Saved amount per false alarm
     Monetary savings by fraud prevented for each false alarm endured (refer to Benchmarking Prevention Performance for details).
   - Intercept
     Number of records hit divided all records fitting training data selection criteria (displayed as "basis points", 100 BP equals 1%).

Rows

Each row represents one indicator for an attribute and the statistical analysis for that indicator. Since all indicators are non-overlapping (that is, any value of an attribute can only be represented by one indicator) the record amounts and numbers can be added when multiple indicators are selected.

Remark

More information can be found on the Automatic and Assisted Rule Generation online help page.
back to top

7.3.6.5 Generated Rules

The bottom section of the rule generation page shows the rules that have already been designed. Notice that you may select multiple rules in this table using the [Ctrl] key and clicking multiple individual rows, and the [Shift] key and clicking an interval of rows. Then you can open a pop-up menu by clicking right on a selected row. You may use this to either save the selected rules to a new or an already existing ruleset, or to delete the selected rules. If you use this to selectively delete rules that you do not want, you can then use the toolbar [Conclude rule generation and save rules] to save the remaining rules to a ruleset.

Notice that if verification is enabled, all respective values in the table are shown underneath the training data value in green.

Toolbar

- [Restart rule generation from scratch]
  Deletes all generated rules and restarts rule generation. This is quicker than stopping and starting rule generation, as IBM Safer Payments can re-use some of the computed data.
- [Conclude rule generation and save rules]
  Enabled if there is any rule in the table. Saves all rules in the table below to a ruleset and concludes rule generation.
- [Conclude rule generation without saving rules]
  Concludes rule generation and discards any results.

Columns

1. Rule
   Name of newly generated rule (the name is created according to the rule generation settings).
2. Condition
   Conditions (one per each line) for this rule.
3. Total
   Total amount (taken from the "amount" meta attribute) and number of all records that are fitting the training data selection criteria defined.
4. Fraud
   Total amount (taken from the "amount" meta attribute) and number of all records marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
5. Genuine
   Total amount (taken from the "amount" meta attribute) and number of all records *not* marked as fraud (taken from "fraud" meta attribute) that are fitting the training data selection criteria defined.
6. Statistics
   Contains various derived performance indicators:
   - Hit rate
     Total fraud amount of this row's data selection divided by the total fraud amount in all records fitting training data selection criteria.
   - False positives
     Number of genuine records falsely hit divided by number of fraud records correctly hit in all records fitting training data selection criteria.
   - Saved amount per false alarm
     Monetary savings by fraud prevented for each false alarm endured (refer to Benchmarking Prevention Performance for details).
   - Intercept
     Number of records hit divided all records fitting training data selection criteria (displayed as "basis points", 100 BP equals 1%).

Rows

Each row represents one rule. Notice that the performance values shown in the right columns are always the incremental performance, i.e. the additional performance with respect to the already existing rules.

Remark

More information on can be found on the Automatic and Assisted Rule Generation online help page.
back to top

7.4 Lists

Lists are used to aggregate data (categorization). A list thus generates a new attribute, which is defined with the respective list. Each list has a number of "values" that are set for the list attribute if the value condition(s) are met. It can either be a text or numeric type attribute.

List data aggregation typically serves two purposes:

- Representation of defined risk using block/allow lists (i.e. known risky merchants, known not risky terminals, etc.). In IBM Safer Payments, this type of list is either represented with lists in a model revision, or as defined risk (defined in the administration section). The difference is that lists in the model revision are changed within revisions by fraud modelling personnel and thus typically represent long-term data, while the defined risk lists can be added and deleted during daily operations by fraud investigation personnel.
- Data aggregation of frequent mappings (MCC, ICA, POS codes, etc.). These types of lists are represented as a specific types of rules that are executed before other rules are executed.

back to top

7.4.1 List

Each list contains the definition of *n* values, where each value can have any number of conditions. The conditions of the values are checked sequentially and once a condition is satisfied, the respective value is assigned to the list attribute.

Notice that value conditions are computed in the sequence the values are displayed in this form (top->down). If the conditions of a value are all satisfied, the value is applied to the output attribute and computation of this list halts. This implies that you may define overlapping list areas. For instance, assume that you define a value "gas station" to the condition "IF MerchantCategory EQUAL_TO 5542" and below the value "shop" to the condition "IF MerchantCategory EQUAL_TO 5000~5999", transactions with the MerchantCategory value of 5542 would be assigned the list output attribute value "gas station", while all other transactions with MerchantCategory values from 5000 to 5999 would be assigned the list output attribute value "shops".

Lists themselves do not require any memory resources, only the new attribute according to its DDC/MDC settings.

back to top

7.4.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.4.3 List Attributes

Each list data aggregation creates exactly one new attribute that it feeds its computational result into.

Each list output attribute is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored of this attribute on disk.
- **Data type**
  IBM Safer Payments supports the data types:
  - **Boolean**
    Stores values of type "true/false", "yes/no" and "1/0" (stored using one Bit). If this attribute is not set by any list value its default value is "false"/"no"/"0". Note: MDC or DDC capacity for boolean type attributes must be a multiple of 8.
  - **Numeric**
    Variable Byte length signed numeric values with variable (0..6) decimals. Both settings determine the universe of the attribute that is calculated in the same form. If this attribute is not set by any list value its default value is "nil" for display and "0" for any computational use (as in a condition). You may also explicitly check this attribute for "nil" (empty) values in conditions.
  - **Text**
    Fixed length text values (configurable length). If this attribute is not set by any list value its default value is "nil" for display and "" (empty string) for any computational use (as in a condition). You may also explicitly check this attribute for "nil" (empty) values in conditions.
  - **Hexadecimal**
    Hexadecimal values (configurable length). Notice that the hexadecimal values can be up to twice as long as the Byte length defined. If this attribute is not set by any list value its default value is "00..." for any computational use (as in a condition) and for display.
  - **IPv4**
    IP address (e.g. 127.0.0.1) values (stored using 4 Bytes). Addresses can be delivered and are displayed as a text of four digit groups (0-255) separated by dots. Internally they are efficiently stored as binary information.
- **Formatted as**
  The formatting options are for display of values on the IBM Safer Payments pages (for examples in queries or case investigation). Choices differ by data type:
  - **Amount**
    Using digit group and decimal separators as defined for each user's preferences (e.g. "12,345.67") for numeric attributes only.

    - **Decimals**
      Using decimal separators as defined for each user's preferences (e.g. "12345.67") for numeric attributes only. This option does not use digit group separators.
    - **ID**
      Using digit group separators as defined for each user's preferences (e.g. "123,456,789") for numeric attributes only.
    - **PAN**
      Using dashed quadruple format typically used for primary account numbers as embossed on cards (e.g. "1234-1243-1243-1243") for numeric and text attributes.
    - **No formatting**
      Shows data with no formatter applied.
- **Length/decimals**
  Quantifies text and numerical data types:
    - **Numeric**
      Byte length of internal storage, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.
    - **Text**
      Byte length of internal storage, with ASCII coded characters, this is exactly the maximum number of characters that can fit into the attribute. Since IBM Safer Payments supports UTF-8 coding, non-ASCII characters may consume multiple bytes. For example, special characters in non-English European languages, such as ä, ü , ö, ß, ê, é, è etc typically require two bytes; all characters of Greek, Cyrillic, Coptic, Armenian, Hebrew, and Arabic require two bytes per character; and Chinese/Japanese /Korean Unified Ideographs require three bytes per character. You thus need to size the byte length of text attribute values according to the UTF-8 character encoding byte space requirements.
- **Unit**
  Displayed with numeric values of this attribute. Typically used for currencies.

back to top

7.5 Indexes

Indexes can be defined with reference to any text, numeric, hexadecimal, or IPv4 type input attribute. They are used to quickly access records that have the same value for the reference attribute as the current transaction message. They are needed for any attribute that identifies a dimension into which past transaction behavior shall be profiled by IBM Safer Payments for real-time decisioning.

Index types

IBM Safer Payments supports three types of indexes: standard, interval, and peer. They all serve different purposes. For example:

- In issuer related fraud prevention, typically cardholder past behavior is profiled, and thus a *standard* index is defined for the PAN (primary account number) attribute.
- In acquirer related fraud prevention, typically merchant/terminal/ATM past behavior is profiled, and thus one or more *standard* indexes are defined for the attributes identifying merchant/terminal/ATM.
- With peer-to-peer payment systems (payment systems where each member can be both a payer and a payee, such as ewallets, online banking, Visa person-to-person, MasterCard MoneySend), the definition of a *peer* index allows for each payer and payee of a transaction to be evaluated with respect to both their payer and payee history.
- To represent standing data that is represented as interval ranges (e.g. BIN/IIN range tables, IP location/intelligence tables), the definition of an *interval* index allows to represent such data with the masterdata capabilities of IBM Safer Payments, and to use them with any transaction.

Notice that you may define multiple (and also different type) indexes for the same attribute. This is useful in rare situations where you for example need evaluation sequences according to different sequence attributes.

All profilings are profiling past behavior and thus are defined alongside an index. This is why in the navigation menu left, these index based profilings are organized under "profilings".

Indexes and sequences

Indexes and sequences are working together to access the history of past transaction records with profilings (and queries). The index itself always points to the most recent transaction record with the respective index attribute value while the sequence for each past transaction record points to the transaction record (with the same index attribute value) *before* this one.

See index sequence help for further details.

back to top

7.5.1 Index

The remainder of this page describes some of the settings of an index and the implications of them. There is also general help on indexes.

Index type

IBM Safer Payments supports different types on indexes that are described in the general help on indexes.

(Source/Target) Attribute

For *standard* and *interval* type indexes, this selection the attribute for which an index shall be created. With *peer* indexes, the source attribute indicates the input attribute that identifies the payer, and the target attribute identifies the payee. Source and target attributes must have the same data type and length.

Notice that you may have multiple indexes for the same attribute (for instance, using a different sequence attribute).

Size and minimum lifetime

Determines how many entries an index can hold simultaneously. Notice:

- Each index is fully stored both in MDC and DDC; there is thus no separate setting of the capacities as with attribute records. (For memory consumption, see below.)
- Once all entries in an index are filled, IBM Safer Payments will overwrite entries that have not been accessed (read or write) within less time than defined as "minimum lifetime". (Notice that there is no special scheme for the overwriting of entries. Entries are overwritten on a "found first" basis.)
- If IBM Safer Payments cannot find an overwritable entry, that is, all existing entries have been accessed last in less than the minimum lifetime, the new entry will not be written and thus discarded. Since this may result in incomplete indexes, this must be considered a severe sizing fault during IBM Safer Payments configuration. IBM Safer Payments generates an event log in the case of an "index overflow". There are special maintenance functions to "re-fill" and index from stored transaction records. Contact the IBM Safer Payments support on their usage. Notice that there are status alarm indicators (SAI) that can alert to fill levels of indexes, which can be useful if you are unsure on how large to size an index.

Resizing

There are two possible cases:

- Enlarge
  Enlarging an index size is always possible. During golive of the revision with the enlarged index, the respective memory sizes are automatically enlarged.
- Reduce
  Reducing an index is only possible when the index area that would be dropped as result of the reduction has not yet been used. This includes outdated entries, so that a reduction is only possible if the index never grew larger than the reduced size. The golive report checks this and only allows a reduced index to go life if these conditions are met.

Purging

Purging allows for the automated removal of all index entries that are older than a defined time period (last read or write access older than maximum lifetime). Purging is performed as part of the "end of day" jobs configured in IBM Safer Payments system configuration. This function for instance is necessary to be used with PCI-DSS compliant applications of IBM Safer Payments. It is also possible to keep track of purged notes in order to speed up the insertion process for new keys in an already filled index.

Memory consumption

Index size determines how many different values can be stored in an index. This size is the same on disk as in memory. It is determined by:

```
(24 Byte + AttributeSize) * IndexSize
```

both for MDC and DDC (*AttributeSize* is 8 Bytes for numeric type attributes and the character length for text type attributes).

Computation conditions

The optional definition of computation conditions allows for the creation of partial indexes, that is, indexes that are only accessed for attribute values of certain transaction messages. Attributes used in these conditions either need to be stored or set by pre-processing rules or lists to allow recomputation of index dependent elements after mergings.

Sequence

The sequence points to the previous transaction record (with respect to the sequence attribute chosen) of each transaction record. Indexes using a sequence are required for counters, mergings, precedents, patterns, collusions and certain queries. See index sequence help for further details.

Insertion conditions

The optional definition of insert conditions allows to have index entries only be generated by specific transaction messages.

back to top

7.5.2 Mergings

Mergings are used to merge multiple transaction messages into one transaction record.

Business background

In payment fraud prevention, a single financial transaction – such as the purchase of goods in a shop – can generate a multitude of different transaction messages. In credit card processing, these typically are:

- **Authorisation request**

A merchant asks an issuer whether they authorise a certain financial transaction. The issuer returns an authorisation code which guarantees the transaction when it is later posted. In some cases, there will be no posting transaction (which causes the actual financial funds transfer) for a granted authorisation. For example, car rental companies frequently use the authorisation to make a "reservation" of funds for potential coverage of damages on their car. When the car is later returned without damage, the authorisation is either revoked or just times out. Any authorisation, regardless of whether or not there is a respective posting, reduces the available balance of the cardholder.

- **Advice**
  Often, IBM Safer Payments is called before the complete authorisation process is finalised. In this case, an authorisation request may be answered differently than the fraud prevention system has recommended. Because the fraud prevention system needs to know about it, typically, a so-called "advice" transaction message is generated that contains information about this and sends it to the fraud prevention system.
- **Posting**
  A merchant posts an actual financial transaction to an issuer. This typically invokes the actual transfer of funds. There are also postings without respective authorisation requests. This is because merchants can also accept financial transactions without previous authorisation. (Whether or not, or to what limit the merchant in this case is guaranteed his payment depends on the individual contract.)
- **Chargeback**
  If the cardholder disputes a transaction with their issuer, the issuer charges back the amount to the acquirer, who in turn charges back the amount to the merchant.
- **Representment**
  If the merchant does not agree with the chargeback, they may represent the transaction to their acquirer, who in turn represents the transaction to the issuer. This starts a manual settlement process of the issue.
- **Fraud**
  If a transaction was deemed to be fraudulent – which can come from multiple sources – a fraud message is generated. This message is also required by most credit card schemes to be delivered to the scheme. Fraud merging requires merging at a later point in time because the fraud alerts typically come in weeks after the actual transactions occurred.

Whilst it is important to differentiate all these transaction messages in payment processing, for the purpose of fraud prevention, they all describe one payment transaction.

Because of this, typically, all these transaction messages are merged into one single transaction within the fraud prevention system. The additional information some transaction messages provide are stored in attributes of the fraud prevention system "transaction record".

Merging process

A complete description of how transactions are processed with IBM Safer Payments is provided on the Message Computation page. Here this process is only described as far as it is relevant to the actual merging process.

Once a transaction message is received in IBM Safer Payments (online or batch), it is first determined which mandator conditions fit for this transaction. Then for each mandator, first lists are computed, then mergings.

Notice that merging involves a merging source and a merging target. The merging source is an incoming transaction message that satisfies the merging source conditions. The merging target is always a transaction record already stored in IBM Safer Payments's MDC/DDC.

Merging consists of the following computational steps:

1. Check merging source conditions on each incoming transaction message for all mandators for which the mandator conditions are met. If the source conditions of a merging are matched, search for matching targets (continue subsequent steps). If the incoming message does *not* fulfill the source conditions of any of the mergings, the transaction is considered *not* to be a merging source and normal computation continues (merging ends here and does nothing until next transaction/record arrives in IBM Safer Payments).
2. To find merging targets, first the value of the attribute of the index of this merging is checked to identify all existing records in IBM Safer Payments MDC/DDC that belong to the same target entity. Potential targets are all records already stored in MDC (and DDC, if the option "Mergings may use DDC" is enabled via the IBM Safer Payments system configuraion and if the respective checkbox is enabled for this merging). They are evaluated back in time with respect to the sequence "timestamp" attribute of the index.
3. Next the (optional) time tolerance criterion ("enforce time" / "tolerance") is checked. Only transaction records that are within the time tolerance specified are considered further as potential targets (again the sequence "timestamp" attribute of the index is used for this comparison). Notice that the time tolerance is symmetrical, that is, the matching source may have a timestamp before or after the matching target.
4. Then the "target conditions" are checked. Only data cache records that satisfy all target conditions will be considered as merging targets.
5. Additionally, termination conditions can be added. The first potential target record that satisfies all defined termination conditions will terminate the computation of the merging. This transaction record is also the last record that is evaluated as a merging target.
6. If exactly one data cache record satisfies all merging target criteria, this record is considered the merging target. Depending on the merging settings, values from the merging source are copied to the merging target and values of the merging target are set to defined values (execution of conclusions). If more than one data cache record satisfies all merging target criteria, the behavior depends on the setting of "Merging method". In its default setting ("first found fitting target record"), only the first record that satisfies all criteria is considered to be the merging target; and only for this merging target, the merging conclusions are executed. In its "all fitting target records" setting, all transaction records satisfying all criteria are merging targets; and in its "closest amount target record" setting, only the data cache record with the closest "amount" meta attribute value is considered to be the merging target.
7. The "store source" option lets you determine under which conditions an incoming transaction message that is identified as a merging source is stored with its input attribute values in IBM Safer Payments's MDC/DDC.
8. If the "re-compute target" option is checked, the target record is recomputed (if there are multiple target records, the latest one with respect to the sequence "timestamp" attribute of the index) and the outputs of this transaction record are passed back as

transaction message responses. If this option is not checked, empty/default values are sent back. This behavior ensures that if for instance you match postings to authorisation transactions, the record is recomputed in case now a reaction should take place.

If a merging succeeded, which means that at least one target was found, then no other mergings will be executed.

The sequence of mergings computed follows the mandator structure from the top (like all other computations in IBM Safer Payments). That is, first the mergings of the top mandator are computed, then the mergings of all mandators down the path to the fitting mandator are computed (if there are any). Within a mandatator's champion model revision, mergings are not computed in a determined sequence.

Merging example

The combination of settings for mergings is rather complex because this feature is very powerful. In most applications, however, the mergings will represent mixing data streams that in your environment must be merged to form a single transaction history that enables fraud pattern detection and fraud management, and this setting will not change after initial configuration of your fraud prevention system.

An example of a merging setting frequently used in credit/debit card fraud prevention is the matching of fraud messages that come from a card management system or other sources. From a business point of view, the authorisation requests and postings that are already in the data cache should be marked fraudulent once a fraud message for the respective transaction comes in.

The first problem frequently faced in this scenario is that fraud messages are generated not by the authorisation system, but by an account management or other separate systems. These systems usually do not share a unique identifier with the authorisation system so that there is no easy way to identify which transaction corresponds to which fraud message. Therefore a specific merging definition must be used.

Typical settings would be:

1. **Source condition**
   Typically, you would have one or more MTID values identifying a fraud message. This would be set as source condition.
2. **Index**
   Usually, you would use the index defined for the "account" meta attribute so that IBM Safer Payments can quickly reduce its search for merging targets to only the transactions of the cardholder. This reduces the search effort significantly as typically only a small fraction of all transactions belong to this one account. Notice that with *peer* type indexes, you also select which index attribute (payer or payee) shall be used, and which sequence (source or target).
3. **Time tolerance**
   Frequently, fraud messages only have the calendar date of the original transaction but not the timestamp. There can also be time shifts due to non-standard conversions or manual processing errors of time zones with international payment transactions. This is why most applications use a time tolerance of 25 hours (or 1.1 days).
4. **Target conditions**
   Here you would include conditions that define which data cache records should be matched to which fraud messages. For instance, "MTID equal_to *POS*;*ATH*" could define that fraud messages should only be merged to records in the data cache that represent authorisation requests or postings. Notice that unlike the source condition, the attribute "MTID" value in the target conditions references the MTID value of the (potential) merging target, not source.
   If you were to perform fraud merging only with the index attribute value and time tolerance criteria, matching would be rather inaccurate. Customers frequently have more than one transaction within 25 hours so that the transaction amount is usually used as another criterion for fraud matching. Because of exchange rate fluctuations and sometimes fees included in transaction message amounts, this criterion is also typically defined with a tolerance. Unlike the time tolerance, however, the amount tolerance is not defined by absolute value but rather by relative value. A typical target condition for this would be "Amount close_to_(by_5%) {Amount}".
   You would also use a target condition to ensure that only currently unmerged records are to be considered as merging targets. A condition "Fraud equal_to 0" ensures that only data cache records not already marked as fraudulent are considered. Fraud messages often also contain MCC and/or MerchantID information. In many applications, however, this value is frequently wrong in fraud messages so that it is rarely used as merging criterion. If in your case, however, you do have such information of good quality, you can include it in the target conditions.
5. **Conclusions**
   All criteria above are used to identify if a transaction message is a merging source and what merging targets exist. The Conclusions determine what happens to the merging target record. Typically, the "fraud" meta attribute would be set to a non-zero value. If a fraud type code exists in the fraud message, you may like to copy this from the fraud message (merging source) to the merging target record in a conclusion "Fraud is {Fraud}".

In this example, you would usually use "closest amount target record" as "merging method" to ensure that if there are multiple merging targets for one merging source, the merging target with the closest "amount" value is chosen. Notice that if the "amount" value difference is outside of the "close_to interval", it will not be considered for merging. If two "amount" values are exactly the same, the least recent data cache record will be selected. This is based on the assumption that if there are two transactions by one customer within a short period of time, for exactly the same amount, they are most likely both fraudulent, so it is more meaningful if the first one is marked as fraudulent. If for the other (later) transaction, a fraud message also exists, it would be matched once the first fraud message is processed.

The store source option lets you define under which circumstances a merging source is stored in the xDC.

The maximum target option defines an upper limit of evaluated target records. If the merging reaches this limit prior to finding a fitting target, the computation is aborted and a log message is created.

The "re-compute target" option enforces the target record to be recomputed after the merging is completed (using the potentially changed record data), and the result of this re-computation is sent back as transaction message response to the merging message request (or in case of a batch file, stored in the log data file). If the option is disabled, the default values of the output attributes are returned with the message response.

back to top

7.5.2.1 Merging

To create a merging definition the following settings are available:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comment**
  Comments are for documentational purposes only. It is advisable to comment the merging fully, so the decision logic remains easy to understand.
- **Index**
  A Merging is defined with an index to quickly reduce the search for merging targets. Notice that with peer type indexes, you also select which index attribute (payer or payee) shall be used, and which sequence (source or target).
- **Store source**
  This setting determines whether or not a transaction which is considered as a merging source is stored within IBM Safer Payments data caches. The following settings are available:
  - **always**
    All merging sources are stored in MDC/DDC.
  - **if no targets found**
    Only merging sources for which no merging targets could be found are stored in MDC/DDC.
  - **never**
    All merging sources are discarded after the merging process has been finished.

  Note:
  In case the merging finds no target record, a subsequent merging might find a target. If that subsequent merging has 'Re-compute target' activated, the source record will not be stored, irrespective of the 'store source' setting.
- **Enforce time**
  If enabled, only transaction records with timestamps (sequence timestamp of the index defined above) within the time tolerance are considered to be potential merging targets (i.e timestamp of merging source message +/- time tolerance).
- **Merging method**
  If more than one data record satisfies all merging target criteria, this setting determines which records are considered to be merging targets. Merging conclusions are executed for all merging targets.
  - **all fitting target records**
    All records that satisfy all target criteria are considered to be merging targets.
  - **closest amount target record**
    Only the record with the closest "amount" meta attribute value is considered to be the merging target.
  - **first found fitting target record**
    Only the first record that satisfies all criteria is considered to be the merging target.
- **Max targets**
  The maximum target option defines an upper limit of evaluated target records within the sequence of the index. If the merging reaches this limit prior to finding a fitting target, the computation is aborted and a log message is created.
- **Re-compute target**
  If enabled, the merging target is recomputed after the merging is completed. The result of this recomputation is sent back as transaction message response to the merging message request (or in case of a batch file, stored in the log data file). If disabled, the default values of the output attributes are returned with the message response. Note that all index computation conditions are evaluated again when a transaction is recomputed. Only if the conditions are satisfied for the target after applying the merging conclusions, index dependent elements will be recomputed.
- **Retry mergings**
  If enabled, then IBM Safer Payments will retry a merging when it doesn't find a target. This can be necessary when you have merging source and target messages that arrive close together while one of the instances is temporarily unavailable. When the instance comes back online, then it is possible that due to the high parallelism of the FLI the source and target messages could arrive on the second instance in the wrong order, and the source would not be able to find the target.
  Note, this is an independent setting from the 'Access Protection' setting under System Configuration -> Serialize Computation. That setting waits until a merging target is completed processing before starting the computation of the merging source and retries the specified number of times. This setting will retry to find a merging target if no target can be found. Note that if both settings are turned on then it can retry the combined amount.
  - **Max Merging Tries**
    When 'Retry Mergings' is enabled, this setting determines how many times it should try
  - **Retry Wait Time**
    When 'Retry Mergings' is enabled, this setting determines how long to wait in between the retries (in msec)
- **Update calendar profiles and events**
  If enabled, calendar profiles and events belonging to the mergings mandator or submandators will get updated, when they meet the mergings conditions and have the option 'Update during merging' enabled.
- **Collusions**
  Selected collusions are triggered when this merging is executed. For further information about collusions please refer to the respective online help page.

back to top

7.5.2.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.2.3 Conclusions

This element uses conclusions. You can find further information in the conclusions chapter:

10.4.2 Conclusions

back to top

7.5.3 Masterdatas

It is often necessary to store certain data that is associated with index attributes. This data could be customer data that can be used with a PAN, merchant data that can be used with a MerchantID, or technical data (for example countries that correspond to BIN ranges). Such data would not be delivered with each transaction, but rather be delivered once and then stored with the respective index attribute (PAN, MerchantID, BIN, etc.).

Such data is referred to within IBM Safer Payments as “masterdata”. Masterdata can be delivered either by “normal” transactions/records that also deliver a “real” transaction/record, or by “special” transactions/records that only deliver the masterdata data (aka “non-monetary” transactions/records). In both cases, a masterdata source transaction/record is identified by the respective “Insertion Condition(s)” of the masterdata definition. Often a specific TrxType or MTID attribute value defines a masterdata source transaction/record. The “Store source” checkbox lets you define whether or not a masterdata source transaction/record shall be included in the data cache or not. Typically if masterdata is delivered by non-monetary transactions, you do not want these transactions to be stored with the data cache.

There is no specific definition of masterdata targets since any transaction/record that is processed after the masterdata source has set a value of a masterdata attribute, this value is taken for processing of this transaction/record.

Remarks

- To change a masterdata attribute value, you just send another transaction/record with the new value. To delete a masterdata attribute value, you just send another transaction/record with empty or zero value. (note that this is true for normal masterdata, but when multi-value masterdata is enabled you need to use the insertion and deletion conditions to update a masterdata)
- If you have multiple masterdata attributes delivered with one transaction/record, you just create multiple masterdata definitions. The storage process of the masterdata attribute values to the index is valid for all masterdata definitions.
- The masterdata attribute must be defined as an input attribute of the model revision (own or inherited).

Once IBM Safer Payments detects a masterdata source transaction message, it extracts the value of the referenced attribute and stores it alongside one referenced index. Masterdata definitions require the sizing of DDC and MDC with the same number of elements as the index to which they refer. The memory requirement for normal (non multi-value masterdata) is thus the referenced index capacity times the masterdata attribute length. See the masterdata page for more information on sizing multi-value masterdata.

back to top

7.5.3.1 Masterdata

To create a masterdata definition, the following settings are available:

- **Name**
  Name of masterdata definition. Often it is best to use the same name as the masterdata attribute selected below.
- **Comment**
  Comments to masterdata definition.
- **Index**
  Masterdata is defined with an index. For instance, if the masterdata definition is about cardholder data, the index that represents the cardholder (e.g. "PAN" or "Issuer" index) must be chosen here.
- **Masterdata attribute**
  Specify the attribute for which the masterdata definition is intended. This must be an own or inherited input attribute. For all transactions for which the insertion condition (below) is fulfilled, the value of the delivering transaction of this attribute is stored in the masterdata cache ("write masterdata"); for all transactions for which the insertion and deletion conditions are not fulfilled, the value of the masterdata cache is written back into the transaction; as if it would have always been part of the transaction ("read masterdata"). When multiple values is enabled (see below) then only the first value from the multi-value list will be written to the transaction. Notice that for *peer* type indexes, the definition of a masterdata attribute for the source and target part of the index is necessary.
- **Store source**
  If checked, the masterdata source message is stored as a record in the IBM Safer Payments data caches. If unchecked, the message is discarded after (the potential) masterdata extraction is carried out. This is only applied to transactions that satisfy the condition defined below.

  Note:
  For the record that is meant to be stored after masterdata was computed a subsequent merging might find a target. If that

merging has 'Re-compute target' activated, the source record will not be stored, irrespective of the 'store source' setting. This can only happen when masterdata and merging source conditions overlap.

- **Allow for multiple values**
  If checked, more than one masterdata value can be stored per index node. Multi valued masterdata can be used in a few different ways. The simple case is just to store multiple values of masterdata which are related to an index. For example, you could store multiple addresses for a customer instead of just their current address. Multi valued masterdata can also be used to enable a multi-relations workflow. To read more about that see the Multi-Relations section.
  When using multiple valued masterdata the multiple values are not used in calculations, due to the potential impact on latency. If the insertion conditions for masterdata are not met and a value is copied from the masterdata cache to a transaction, then it will always be the first value in the multi value list. There is currently no way to select a different value to be written or to not write a value to the transaction. This may result in confusion when examining the resulting transactions if it does not make sense that the first value in the list should be applied to the transaction.
- **Capacity for multiple values**
  The capacity for multiple values is the overall number of additional elements stored. This is the capacity for all elements in the multiple values storage, so for example, if you had an index capacity of 10000 elements, and wanted to store up to 5 elements per index node, you would set the capacity to 40000. This would not mean that you are limited to 5 index entries per node, it is possible to store 20 entries on some nodes and only 1 entry on other nodes, but the full capacity (the sum of all the nodes) is allocated here.
- **Associated index**
  The index which will be used to find relations in a multi-relations workflow. When multiple value masterdata is displayed in a case then this index will be searched using the values given in the multiple valued masterdata. For more information see the Multi-Relations section. Note that this index is only used to help display more information in user queries, it is not used in computations. If you do not select an index then Safer Payments will attempt to find the appropriate index based on the attribute which was selected.
- **Relationship values**
  Attributes used to describe the relationship between each index node and each masterdata value associated with it. When you have multiple values enabled for masterdata then you may want to add extra information about the relationship between the index node and the particular masterdata value. For example, if you are storing multiple addresses for a customer, you may want to also store the information about which one is the current address, which can be stored in a relationship attribute (although, note that in this example, you would also need to update the relationship description when a new address was added)
  Relationship attributes need to be passed in the same transaction as the masterdata value they are related to and are only updated if the insertion conditions for the masterdata are hit. Relationship attributes will always be updated when that masterdata value is sent, so if you send empty relationship values with an existing masterdata value then that will effectively delete the existing relationship values.
- **Insertion conditions**
  Insertion conditions decide if the masterdata cache value (the value that "sticks") is written from the transaction to masterdata cache or read from masterdata cache to the transaction. When multiple values are not enabled then if the insertion conditions are hit then it will take the masterdata attribute value from the incoming transaction and overwrite the existing masterdata cache value in memory (so if an empty value is sent then it will effectively delete the existing masterdata value). When multiple values are enabled, then when the insertion condition is hit then it searches for the value from masterdata attribute of the incoming transaction in the existing list of multiple values. If the value from the incoming transaction is not found in the list, then it is added to the end of the list along with any relationship attributes. If the value is found then only the relationship attributes will be updated.
- **Deletion conditions**
  Deletion conditions allow you to delete values from the masterdata when using multiple values (although they can also be used for normal masterdata, it may be simpler to just send empty values which will overwrite the existing values). When a deletion condition is hit, then Safer Payments will search the existing list of masterdata for the appropriate node and if it finds a matching masterdata value, then it will remove it, along with any associated relationship information. You should generally attempt to use non-overlapping conditions for your insertion and deletion conditions, but if both are hit, then the insertion will take precedence and the deletion will not be performed.

Memory consumption

The size of masterdata is determined by a combination of index capacity and multiple values capacity (if you have multiple values enabled). This size is the same on disk as in memory (except when deferred writing or encryption are enabled). The base masterdata size (without multiple values enabled) is determined by:

```
AttributeLength * IndexCapacity
```

both for MDC and DDC (*AttributeLength* is configurable in the input attribute setup). When multiple values are turned on the sizing of the masterdata is more complex. If you turn multiple values on, then in addition to the base masterdata size above it will also allocate:

```
((8 Bytes) * IndexCapacity) + ((8 Bytes + AttributeLength) * MultipleValuesCapacity)
```

for MDC and DDC. In addition, if masterdata attribute is of type Hex or IP, then an additional (1 bit * *IndexCapacity*) will be required.

Each relationship attribute also requires its own memory. Every relationship attribute you add requires:

```
(RelationshipAttributeLength * IndexCapacity) + (RelationshipAttributeLength * MultipleValuesCapacity)
```

back to top

7.5.3.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.4 Device Identification

IBM Safer Payments device identification keeps histories of up to 4 devices per counterparty (user account). For each of these devices (browsers), the following data is computed and stored:

- number of transactions with this device
- earliest and latest transaction date
- number of cookie and device ID mismatches

This information is made available from within the device identification profiling profiling method. For each device identification, output attributes carrying this information can be defined.

Notice that you can define multiple device identifications, for instance to identify customer and merchant devices.

back to top

7.5.4.1 Device Identification

To create a device identification definition, the following settings are available:

- **Name**
  The name of a device identification definition. It is used in all IBM Safer Payments forms.
- **Comment**
  A description of the device identification. This will be displayed in various forms and allows to add a more verbose description of the device identification.
- **Index**
  Device identification is defined with an index. For instance, if the device identification definition is meant to match devices in online banking, this would be an index on account number. If a peer index is selected, the device identification offers the additional computation method "last target fingerprint".
- **Timestamp**
  An attribute that contains timestamp information. This timestamp should be the point in time when the device identification has been gathered on the end-user device.
- **Time unit**
  The time unit that will be applied to computation outputs.
- **Cookie**
  An attribute that contains four byte of hexadecimal cookie data. This can be the result of the crc32 hash transformation or any other four byte hexadecimal attribute. As long as this value does not change between two transactions the device identification will consider two devices to be the same.
- **Fingerprint**
  Another attribute that contains four byte of hexadecimal fingerprint data. This can be the result of the crc32 hash transformation or any other four byte hexadecimal attribute except the one used as "Cookie". As long as this value does not change between two transactions the device identification will consider two devices to be the same even when the cookie has changed.
- **Computation Conditions**
  Computation conditions allow to define filters on incoming transaction data. Only transactions that match all defined conditions are taken into account for the device identification.
- **Output attributes**
  Every device identification can compute one or more output attributes. For detailed information refer to the respective help text.

Memory consumption

Device identifications are defined with reference to an index. Their memory consumption is computed as:

```
80 Bytes * IndexSize
```

both for MDC and DDC. The memory consumption for all computed output attributes has to be added.

back to top

7.5.4.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.4.3 Device Identification Attributes

Each device identification profiling creates one or more new attributes that it feeds its computational results into. Notice that if you use a mandator structure, the rules of this decision logic may use all attributes defined in champion mandator revisions above it in the structure. Each of the device identification output attributes is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain. Notice that the attribute names do not need to correspond to the variable names of data delivered to IBM Safer Payments; you define the relation between IBM Safer Payments attributes and variable names in "Mappings".
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored for this attribute on disk.
- **Data type**
  Computations (see below) either deliver numeric, boolean, or hexadecimal output depending on the chosen computation method. The formatting and length of output attributes can only be adjusted for computations delivering numeric output.
- **Formatted as**
  The formatting options are for display of values on the IBM Safer Payments pages (for examples in queries or case investigation). Choices differ by data type:
  - **Amount**
    Using digit group and decimal separators as defined for each user's preferences (e.g. "12,345.67") for numeric attributes only.
  - **Decimals**
    Using decimal separators as defined for each user's preferences (e.g. "12345.67") for numeric attributes only. This option does not use digit group separators.
  - **ID**
    Using digit group separators as defined for each user's preferences (e.g. "123,456,789") for numeric attributes only.
  - **PAN**
    Using dashed quadruple format typically used for primary account numbers as embossed on cards (e.g. "1234-1243-1243-1243") for numeric and text attributes.
  - **No formatting**
    Shows data with no formatter applied.
- **Length/decimals**
  Byte length of internal storage for numeric output attribute values, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.
- **Unit**
  Displayed with numeric values of this attribute. For all computations that work on timestamps or durations this is set to the time unit defined in the device identification definition.
- **Computation**
  This setting determines how the output attribute is computed from the set of transaction records selected by this device identification:
  - **Average number of uses per time unit**
    Computes the average number of uses per time unit for the matched device. If no device is matched and a new device created, the output value is (0).
  - **Cookie changed**
    Indicates that a device uses an unknown cookie. If the fingerprint attribute matches, the new cookie is stored and the device identification information is kept. If the fingerprint also changed a new device is created (and an old device is potentially overwritten).
  - **Fingerprint changed**
    Indicates that a device uses an unknown fingerprint. If the cookie attribute matches, the new fingerprint is stored and the device identification information is kept. If the cookie also changed a new device is created (and an old device is potentially overwritten).
  - **Last target fingerprint**
    This computation method is only available in combination with a peer index: It computes the fingerprint of the last device used with the target account of the current transaction.
  - **Number of cookie mismatches**
    Counts the number of times the cookie has been changed for a given device. The number is incremented, whenever a device is matched only by the fingerprint.
  - **Number of fingerprint mismatches**
    Counts the number of times the fingeprint has been changed for a given device. The number is incremented, whenever a device is matched by the device cookie but not by the fingerprint. This could happen e.g. when the device configuration

changes.

  - **Number of uses**
    The number of transactions that have been matched to the device.
  - **Time between first and previous use**
    The time between the initial use and the second most recent use of a device with respect to the timestamp attribute.
  - **Time since first use**
    The time between the initial and the most recent use of a device with respect to the timestamp attribute.
  - **Time since previous use**
    The time between the second most recent use and the most recent use of a device.

Examples:

- The following table shows a sequence of transactions together with the computed outputs:

#	Account (Index)	Cookie	Fingerprint	Number of uses	Number of cookie mismatches	Number of fingerprint mismatches	Cookie changed	Fingerprint changed
1	1234	0xAAAAAAAA	0xCCCCCCCC	1	0	0	Yes	Yes
2	1234	0xAAAAAAAA	0xCCCCCCCD	2	0	1	No	Yes
3	4321	0xBBBBBBBB	0xDDDDDDDD	1	0	0	Yes	Yes
4	1234	0xAAAAAAAA	0xCCCCCCCD	3	0	1	No	No
5	1234	0xAAAAAAAB	0xCCCCCCCD	4	1	1	Yes	No
6	4321	0xBBBBBBBB	0xDDDDDDDD	2	0	0	No	No
7	1234	0xAAAAAAAC	0xCCCCCCCC	1	0	0	Yes	Yes
8	1234	0xAAAAAAAB	0xCCCCCCCD	5	1	1	No	No

- **Remarks:**
  - Please not that #3 and #6 are transactions for a different account.
  - As long as either the cookie or the fingerprint of a device matches, the device identification is updated with data from the current transaction. The cookie is always matched first.
  - In #7 a new device is created because neither cookie nor fingerprint match the previous transaction for this account. As every device identification has slots for up to four devices, the data from #5 is still available and #8 can be matched to it.

- The following table shows a sequence of transactions together with the computed outputs for time unit hours:

#	Account (Index)	Cookie	Fingerprint	Timestamp	Time since first use	Time since previous use	Time between first and previous use	Average number of uses per time unit
1	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 00:00:00	0.00	0.00	0.00	(0)
2	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 01:00:00	1.00	0.00	0.00	2.00
3	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 03:00:00	3.00	2.00	1.00	1.33
4	1234	0xBBBBBBBB	0xDDDDDDDD	1970-01-01 06:00:00	0.00	0.00	0.00	(0)
5	1234	0xAAAAAAAA	0xCCCCCCCD	1970-01-01 09:00:00	9.00	6.00	3.00	0.44
6	1234	0xAAAAAAAA	0xCCCCCCCD	1970-01-01 12:00:00	12.00	3.00	9.00	0.42

- **Remarks:**
  - The transaction in #4 affects the same account but is matched to a new device because neither cookie nor fingerprint match.
  - Transactions #5 can be matched to the first device via the cookie. #6 is also matched to this device by cookie and fingerprint.

- The following table shows a sequence of transactions together with the computed outputs:

#	Source Account (Index)	Target Account (Index)	Cookie	Fingerprint	Last target fingerprint
1	1234	0000	0xAAAAAAAA	0xCCCCCCCC	0x00000000
2	5678	0000	0xBBBBBBBB	0xDDDDDDDD	0x00000000
3	1234	5678	0xAAAAAAAA	0xCCCCCCCC	0xDDDDDDDD

- **Remarks:**
  - The "last target attribute" computation is only available for peer indexes.
  - The output attribute is set to the last known fingerprint value for the target account. If the target account does not have any fingerprint set from a previous transaction, the output is "0x00000000". This computation never changes the device identification for the target account.

- The following table shows an example with transactions for one device that are coming in out of sequence:

#	Account (Index)	Cookie	Fingerprint	Timestamp	Number of uses	Time since first use	Time since previous use	Time between first and previous use	Average number of uses per time unit
1	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 01:00:00	1	0.00	0.00	0.00	(0)
2	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 03:00:00	2	2.00	0.00	0.00	1.00
3	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 00:00:00	3	3.00	2.00	1.00	1.00
4	1234	0xAAAAAAAA	0xCCCCCCCC	1970-01-01 06:00:00	4	6.00	3.00	3.00	0.67

- **Remark:**

The computed output values for out-of-sequence transactions will always be calculated with respect to the most current transaction (with respect to the timestamp attribute). Therefore the "number of uses" in transaction #3 is "3".

back to top

7.5.5 Precedents

Precedents allow quick and easy access to values of the previous transaction record's value by copying it into an attribute of this transaction message.

This for instance allows to define for attributes in this transaction message:

- Amount of last transaction of this cardholder.
- Amount of last transaction at this merchant.
- Amount of foreign last transaction of this cardholder.
- Amount of last transaction of this cardholder above $200.

back to top

7.5.5.1 Precedent

A precedent copies the value of the "source attribute" of a previous transaction record:

- of the most recent transaction record in the index dimension chosen,
- that satisfies the conditions (if defined)

and stores them into its output attribute.

Index

IBM Safer Payments searches for the preceeding record in the dimension of an index sequence. Only indexes that have sequences can be selected here. Notice that with *peer* indexes, the preceeding record can be searched for in either the "direction" of the source (payer) or the target (payee) sequence.

Source attribute

Lets you select which attribute's value should be captured (and outputted) by this precedent. Notice that the attribute type/length of the output attribute of a precedent is exactly the type/length of this source attribute.

Max records

Maximum number of records alongside the sequence that are evaluated into the past. You should define a reasonable limit here so that in case of very long sequences and a precedent not being found, computation times do not get too high.

Include DDC

Check to have IBM Safer Payments include data from disk with this computation. Notice that this may significantly increase computation time.

Computation Conditions

The first record into the past (that is, back in sequence time from now into the past) that satisfies this/these computation condition(s) is considered the precedent. The source attribute value of this record will be taken as output attribute of this precedent and is passed to the decision model with this transaction message.

back to top

7.5.5.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.5.3 Precedents Attributes

The output attribute of a precedent is of the same type, format, and length (and if numeric, same decimals) as the source attribute defined above. The required type/length(/decimals) settings are thus taken automatically from the source attribute and are not shown or editable in this form. If categories are defined for the source attribute, they will be assigned to the output attribute of the precedent automatically. In this case the categories are shown at the bottom of the page but are not editable.

It, however, typically has a different name and sometimes different storage settings.

The settings of a precedent output attribute are hence:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Profiling generated attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored for this attribute on disk.

back to top

7.5.6 Calendar Profiles

Calendar profiles provide aggregated counts of totals and transaction message frequencies over calendar periods and store these alongside a referenced index. The number of calendar periods held according to the transaction message timestamp sequence can be set freely. The periods are based on the values of the meta attribute "timestamp" of the transaction message sequence. Each timestamp later than the last one received, moves "transaction time" forward. This transaction time is the reference for the calendar periods.

Within a calendar profile, new attributes can be defined that carry totals or frequencies from any calendar period tracked by the profile.

Memory consumption for the bare calendar profile is computed as:

(12 Bytes * *NumberOfPeriods*) * *IndexSize*

When "support standard deviation" is enabled the memory consumption will be:

(20 Bytes * *NumberOfPeriods*) * *IndexSize*

both for DDC and MDC. In addition the new attributes memory consumption must be added.

Computation of calendar profiles is a two-step task ("pre" and "post"). If the profile shall include the current transaction, only input attributes and list/precedent profiling output attributes can be used in its condition ("pre"). If the profile is defined to not include the current transaction, all other attributes can be used as well in its conditions ("post"). Because the "other attributes" are computed after the profile was (pre-)updated, in this case, a (post-)update step takes place after the rules are computed. This allows defining profiles on rule output attributes, for instance such as a profile for transaction message intercepts or case generation alerts.

back to top

7.5.6.1 Calendar Profile

For details on calendar profile definition, rest the mouse pointer over the respective field to show tooltip style explanations and read below.

Each calendar profile is specified by a set of definitions that are entered on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are for documentational purposes only. It is advisable to comment the calendar profile fully, so the decision logic remains easy to understand.
- **Index**
  Each calendar profile is computed with an index. For example, to profile cardholder behaviour, you would use an index on the primary account number; to profile merchant behavior, you would use an index on the merchant ID. Notice that the index chosen does not need to have a sequence. If you selected a *peer* type index, you may also select which index attribute (payer or payee) shall be used for profiling.
- **Computation timestamp**
  Determines whether the reference time of the profile (explained below) or the transaction time is used to compute the output attributes of the profile. If set to "Reference time" period 0 is considered as the reference time (most recent timestamp that has been seen in the meta attribute "timestamp"). If set to "use other" period 0 is considered as the transaction time in the selected timestamp attribute.
- **Include current**
  If checked, the current transaction message data (including everything computed) is used in the calendar profile's value.
- **Update during merging**
  Update calendar profile when a merging with setting `update calendar profiles and events` or a manual fraud mark has changed the profile's conditions. Example:

- Transaction 1 amount: 100. Calendar total amount: 100
- Transaction 2 amount: 30. Calendar total amount: 100+30=130
- Merging changes the amounts of transaction 1 and 2 to 10
- Transaction 3 amount: 300. Calendar total amount: 10+10+300=320

Note: Make sure to activate `update calendar profiles and events` in all mergings that have conclusions which use the calendar profile's amount attribute or conditions attributes. If there are some of those mergings that do have the option enabled and others that do not, it can lead to unexpected values in the profile's output attribute(s).

- **Support standard deviation**

  When enabled, standard deviation can be chosen as an output attribute's computation. Note that this will change the memory consumption for the bare calendar profile to:

  ```
  (20 Bytes * NumberOfPeriods) * IndexSize
  ```

  Changing this value will delete all calendar period data after golive.

- **Decimal accuracy**

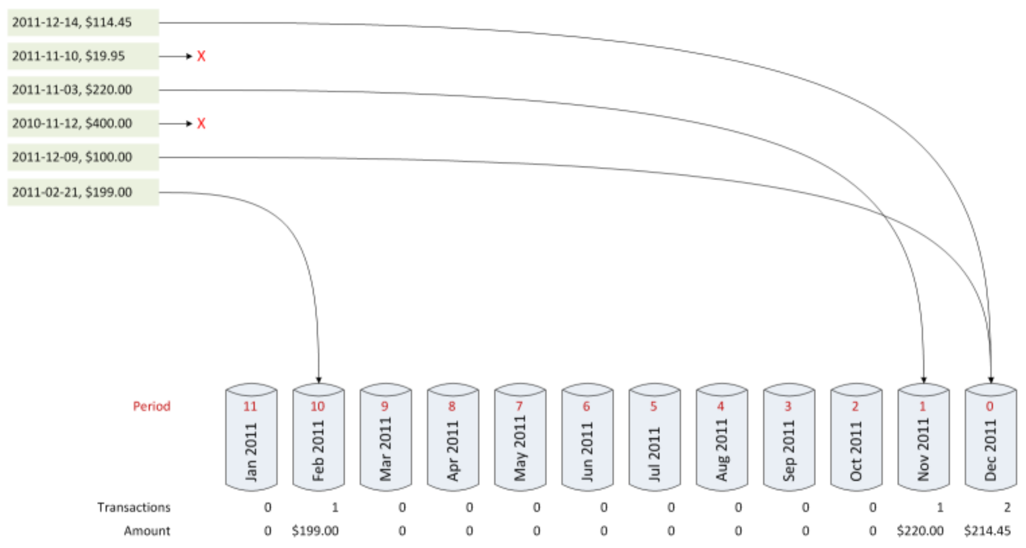
  Sets the decimal accuracy for standard deviation. The larger the value, the more precise standard deviation can be calculated. In principle, this is only necessary when the deviation output attribute shows more decimals than the amount attribute uses. Enlarging the value comes at a price, since very large values need to be saved for computing deviation. Enlarging the accuracy enhances the probability that the values cannot be saved in the 8 bytes integer anymore. Setting the accuracy to "0" is good enough for most requirements.

  Changing this value will delete all calendar period data after golive.

- **Amount attribute**
  While the default of this is the amount meta attribute defined, you may specify an alternative attribute to compute the calendar profile's amount values (for instance, another currency).
- **Time zone offset**
  Profiles are strictly calendrical. In order to decide to which calendar period a transaction message is added, profiles use the timestamp meta attribute of the model revision. If the transaction message's timestamp meta attribute values are in a different time zone than the periods of the profile, you can enter a non-zero value here. This value in seconds is added to the transaction message's timestamp meta attribute value before the fitting period is computed.
- **Calendar period**
  Profiles are strictly calendrical and so are the calendar periods.
- **Number of periods**
  Number of calendar periods that are considered by the profile (including the current period).
- **Exclude zero periods**
  If checked, periods which have no fitting transaction, will be excluded from computation.
- **Computation Conditions**
  Determine which transaction messages are included in the calendar profile.
- **Output attributes**
  You may define one or more attributes that deliver calendar profile data to be used in subsequent decision logic elements.

Example

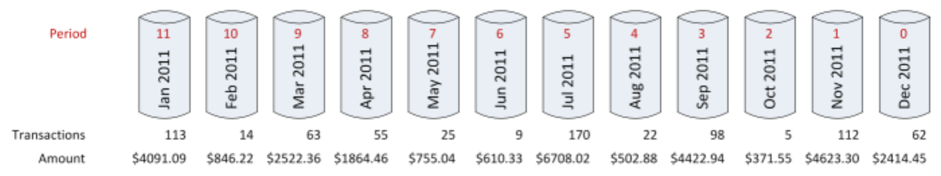
Assuming a calendar profile is defined that counts all transaction messages with an amount larger than or equal to $100.00 over calendar periods of 12 consecutive months, the figures below show how some sample transactions are stored in a profile (assumed today is the 2011-12-14):



The green boxes depict transaction with their transaction timestamp and their amount. Notice that the transaction timestamp denotes when the transaction actually occurred at the point of sales (meta attribute "timestamp"); this is not the system timestamp. The 12 gray containers depict the 12 calendar periods for which the profile collects data. The periods are numbered from "0" (this calendar period) to "11" (oldest recorded period). Now six example transactions come in:

1. Counted in calendar period 0
2. Not counted because profile condition ("amount ≥ $100.00") not met
3. Counted in calendar period 1
4. Not counted because transaction timestamp is before the profile's calendar periods
5. Counted in calendar period 0
6. Counted in calendar period 10

To exemplify how output attribute values are computed for this, assume profile calendar periods more filled:



Notice that you may define any number of output attributes that are computed differently from the same calendar periods. Assuming that you want to compare last three months' performance to the performance of the three months before. In this case, you would define an output attribute with "past calendar period(s)" of "0~2/3~5" (most recent period first in intervals). This would compute for number of transactions as:

```
(62 + 112 + 5) / (98 + 22 + 170) = 179 / 290 = 0.6172
```

Notice that when you define a ratio (such as above), you would probably want to define an output attribute with decimals.

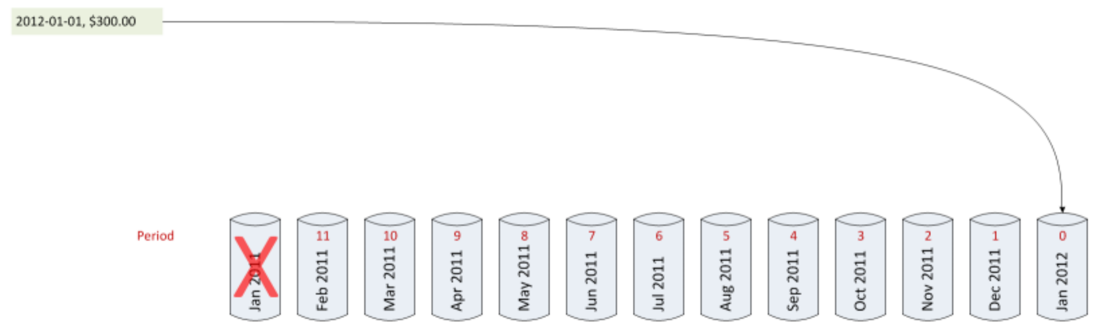
For the computation of total amounts the computation would be:

```
(2414.45 + 4623.30 + 371.55) / (4422.94 + 502.88 + 6708.02) = 7409.30 / 11633.84 = 0.6369
```

Because profiles are strictly calendrical, once a new calendar period starts, it starts empty and is only filled during the period. If this is not wanted, IBM Safer Payments can project the new period value (if "number of periods" is greater than one). In order to do so, enable the "projection" function in a profile output attribute (more details on profile output attribute online help page). In this case, the above number of transaction computation for "past calendar period(s)" of "0~2/3~5" would be:

```
((((62 * 14) + (112 * (31 - 14))) / 31) + 112 + 5) / (98 + 22 + 170) = 0.7118
```

As time passes, eventually transaction messages come in that are past the most recent calendar period. In this case the profile performs a so-called "rollover", that is, the oldest period is discarded and a new period is created:



Notice that if the transaction would not have been January 2012 but March 2012, the profile would rollover three calendar periods at once. This implies that if you would have a "freak transaction", that for instance references a timestamp meta attribute value that is years into the future, the profile would rollover all periods and thus delete all periods.

In certain applications, transactions are not sent to IBM Safer Payments in ordered sequence with respect to the point-of-sales timestamps (timestamp meta attribute). This is for instance the case, if you feed authorisation requests and posted transactions into IBM Safer Payments. Because authorisation requests are real-time and posted transactions are often nightly batch feeds, transactions could come in that are for an earlier point in time than the newest transactions that had arrived in IBM Safer Payments until this point in time. IBM Safer Payments keeps track of the newest transaction's timestamp and is also known as the "reference timestamp". This has another use when it comes to the computation of the profile output attributes. For the computation of these attributes, you may select if this should consider "transaction time" or "reference time" (setting "computation timestamp"). For transactions that come in sequence (each timestamp meta attribute value of the subsequent transaction message is greater than or equal to the one of the transaction before), this setting makes no difference since the "reference time" is the same as the "transaction time". However, for an "out-of-sequence" transaction, such as a posted transaction that comes in a few hours after it was made at the point of sales, this setting does make a difference. If set to reference timestamp, computation of the said transaction uses the profile values (i.e. computes the profile's output attributes) according to the reference timestamp which is the newest timestamp that has been seen by Safer Payments (possibly considering transactions that were made after the current one but arrived before). This means that a period is considered a certain back period based on the reference timestamp. If set to another timestamp attribute, computation of the said transaction uses the profile values according to the value of the selected timestamp attribute (e.g. meta attribute timestamp) of the transaction. This means that a period is considered a certain back period based on the timestamp of the transaction.

If the most recent transaction received by IBM Safer Payments had the timestamp meta attribute value of 2011-12-14, and a new transaction comes in with timestamp meta attribute value 2011-10-20, reference time computation would compute calendar profiles exactly as exemplified above. Transaction time computation, however, would re-create the result as follows:

```
(5 + 98 + 22) / (170 + 9 + 25) = 0.6127
```

Notice that for past calendar periods, no projection is used and the full value of the month of October was used.

Remarks

- When you enlarge the number of profile calendar periods into the past, the "new" periods will initially not be filled but only fill up with the passage of time.
- When you decrease the number of profile calendar periods into the past, the periods not used anymore will be permanently deleted.
- When computing profiles for sandbox records, no rollovers are performed. This means when testing old periods are not discarded but kept and still included in the computation.

back to top

7.5.6.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.6.3 Calendar Profile Attributes

Each calendar profile creates one or more new attributes that it feeds its computational results into.

Each of the calendar profile output attributes is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Profiling generated attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored for this attribute on disk.
- **Data type**
  Calendar profile output attributes are either frequencies (numbers) of records or amounts. Thus only the numeric data type exists. Frequently frequencies are expressed by length 2 attributes, and amounts by length 4 / decimals 2 attributes.
- **Formatted as**
  The formatting options are for display of values on the IBM Safer Payments pages (for examples in queries or case investigation). Choices are:
  - **Amount**
    Using digit group and decimal separators as defined for each user's preferences (e.g. "12,345.67").
  - **Decimals**
    Using decimal separators as defined for each user's preferences (e.g. "12345.67") for numeric attributes only. This option does not use digit group separators.
  - **ID**
    Using digit group separators as defined for each user's preferences (e.g. "123,456,789").
  - **PAN**
    Using dashed quadruple format typically used for primary account numbers as embossed on cards (e.g. "1234-1243-1243-1243").
  - **No formatting**
    Shows data with no formatter applied.
- **Length/decimals**
  Byte length of internal storage, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.
- **Unit**
  Displayed with values of this attribute. Typically used for currencies.

- **Computation**
  Determines how the attribute shall be computed from the calendar profile.
- **Past calendar period(s)**
  Selects how the value of this attribute is to be computed from the past calendar periods defined. Periods are denoted by numbers: the value "0" corresponds to the current period, the value "1" to the first past period, and the value "(*n*-1)" corresponds to the oldest period (*n* is the "number of periods" as set with the calendar profile definition above). There are multiple types of computation you can define here:
  - **n**
    If you just specify a single number, the value of exactly this calendar period is applied to this attribute.
  - **n~m**
    If you specify a number interval, the total value (sum) of all calendar periods of this interval is applied to this attribute. Notice that this interval is inclusive, for instance, if you define "3~5", the profile calendar period is "monthly" and the current transaction is in mid-December, the value applied to the profile output attribute would include the past months July, August, and September of this year.
  - **min(n~m)**
    Computes the minimum of the calendar period values in the specified interval.
  - **max(n~m)**
    Computes the maximum of the calendar period values in the specified interval.
  - **avg(n~m)**
    Computes the average of the calendar period values in the specified interval.
  - **(n~m)/(k~l)**
    Computes the ratio of the sum of the interval "n~m" divided by "k~l" (intervals computed as above). Also supports entries such as "n/k~l)", "(n~m)/k", and "n/k" (brackets are ignored and only put here for illustration). Notice that this value is not in percent, but absolute.
- **Projection**

  If checked, the value of the current calendar period (value "0" in "past calendar period(s)" settings above) is computed as projection in respect to the last period.

  Projection is calculated according to formula

  ```
  (valueLastPeriod * ((periodLength - timeSinceCurrentPeriodBegin) / periodLength)) + (valueCurrentPeriod *
  (timeSinceCurrentPeriodBegin / periodLength))
  ```

  The option is only available, if the "number of periods" as set with the calendar profile definition above is above "1". Notice that when the period "0" appears in intervals or ratios, and "projection" is checked, the current period is also computed as projection.

  The algorithm takes into account the fact that calendar months, quarters and years are not of uniform length. It will use the respective period’s true lengths for the projection.

  Projection is not supported for standard deviation computation.

back to top

7.5.7 Patterns

Business background

Certain fraud schemes involve a certain sequence of transactions. For instance, after one international purchase transaction, there would be a domestic purchase transaction, and immediately after that domestic transaction, an international ATM withdrawal follows. To detect such schemes, IBM Safer Payments provides “pattern” profiling.

For details, see online help pattern.

back to top

7.5.7.1 Pattern

For the business background, see online help patterns.

Definition

Pattern definition consists of:

- conditions that identify when pattern profiling is performed,
- stencils that describe each part of a pattern, and
- an output attribute generated by pattern profiling, indicating the pattern being matched with the current transaction message.

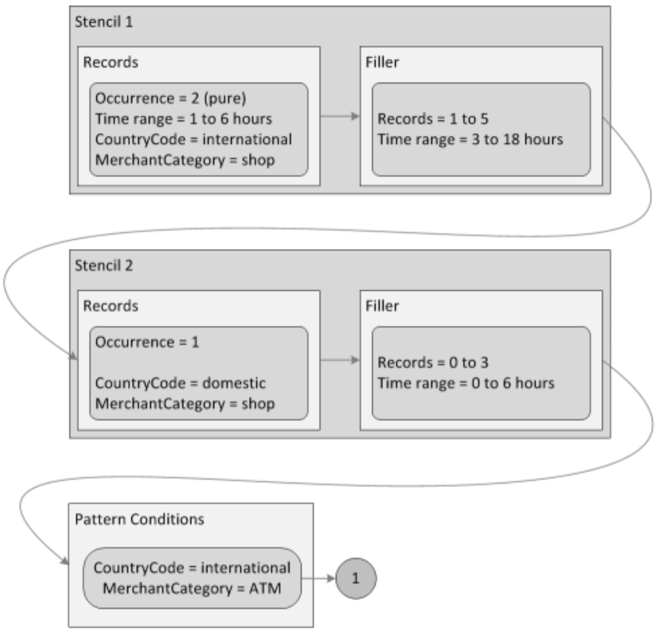
One set of conditions is applied to each incoming transaction message processed by IBM Safer Payments. Only if all conditions are met, IBM Safer Payments will evaluate if the past transactions history of this transactions message matches the pattern defined.

Each stencil is a combination of another set of conditions that a past transaction record shall satisfy, and a range of “filler” transaction messages (or a “filler time range”) that may occur after the condition matching record had occurred. All stencils defined must be satisfied by the transaction record sequence in order for the pattern to be matched.

The output attribute is fixed of type "boolean" (True/False, hereinafter "1"/"0"), and of computation "occurs". Pattern profiling only assigns values "0" and "1" to it. "0" indicates that either the condition was not matched by the current transaction message or that the stencil sequence was not matched by previous transaction records. "1" indicates that both the condition was matched for the current transaction message and that the stencil sequence was matched.

Example

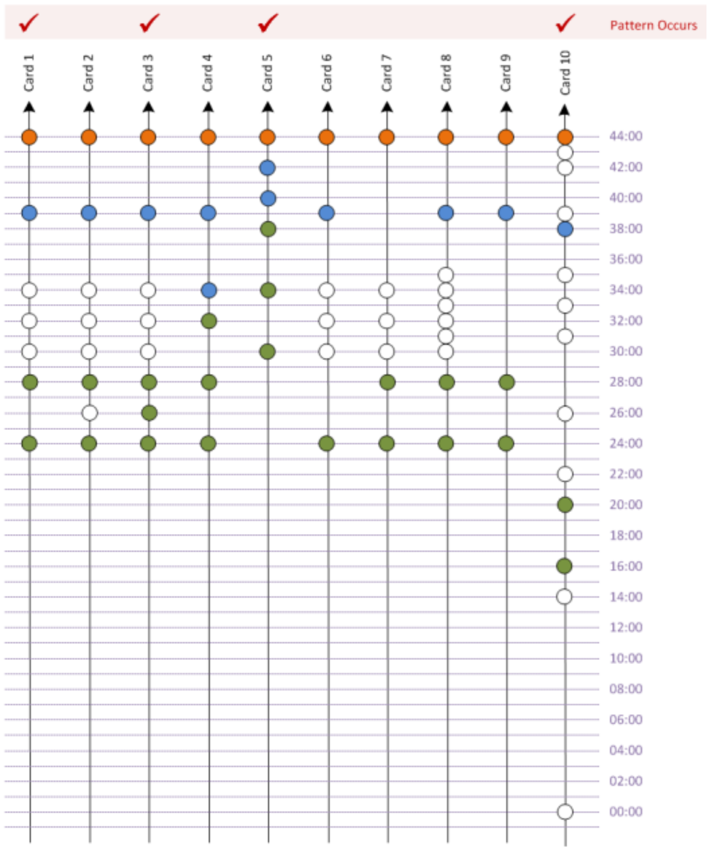
The example assumes this pattern definition:



In this example, the pattern output attribute is assigned the value of "1" if:

- There are two subsequent ("pure") international purchase record that are minimum 1 hour and maximum 6 hours apart (stencil 1 / records).
- Thereafter there are minimum 1 and maximum 5 records within minimum 3 and maximum 18 hours after the past international purchase record that do not satisfy the records condition of stencil 2 of a domestic purchase record (stencil 1 / filler).
- After these filler transactions, but also within maximum 18 hours, there is one domestic purchase record (stencil 2 / records).
- Thereafter there are up to maximum 3 records of any kind plus the current international ATM transaction (pattern conditions) which have to occur within maximum 6 hours (stencil 2 / filler).

The following figure uses a number of sample sequences to exemplify how patterns are computed:



Timelines for each example card with their transaction sequences are drawn on a vertical arrow. Time moves from bottom to top. For illustration reasons, all pattern condition matching – and thus computation triggering transaction messages – are positioned all at 44:00h.

Dots mark transactions on the example cards timelines:

- Red: current transaction message satisfying the pattern conditions.
- Blue: past transaction record satisfying stencil 2 conditions.
- Green: past transaction record satisfying stencil 1 conditions.
- White: past transaction record that does **not** satisfy the current or next stencil's conditions.

The “pattern occurs” value of the output attribute for the example transaction sequences would be computed as:

1. There are two (green) transaction records satisfying stencil 1 records’ conditions. Thereafter, there are three (white) transaction records not satisfying stencil 2 records’ conditions; these transaction records are within the stencil 1 filler conditions of number 1 to 5 (there are 3 transaction records) and time range 3 to 18 hours (11 hours) before the (blue) first stencil 2 records’ condition satisfying transaction record. Finally, the current transaction message satisfied the pattern conditions. The pattern occurs value hence is “1”.
2. Different to card 1, in this example, there is one (white) not stencil 1 records condition satisfying transaction record between the two (green) ones satisfying it. Because the “pure” option is enabled for the stencil 1 records definition, the non-satisfying transaction record causes stencil 1 to not apply and hence the pattern occurs value is “0”. Notice that if the “pure” option would not be enabled, the pattern occurs value would be “1”.
3. Same as card 1, only that there are three rather than two (green) stencil 1 records condition satisfying transaction records subsequent to each other. Since the first two transaction records already satisfy the stencil 1 records condition, the third (green) one is counted as a filler, and since 4 filler transaction records are also within the number 1 to 5, and the time distance between the last stencil 1 record and the first stencil 2 record is 13 hours which is within the time range 3 to 18 hours. The pattern occurs value hence is “1”.
4. The first two (green) transaction records satisfy stencil 1 records condition, thus the third (green) transaction record, even though also satisfying this condition is considered to be the first stencil 1 filler transaction record. The next (blue) transaction record satisfies stencil 2 records condition, thus the next (blue) stencil 2 records condition satisfying transaction record is considered the first stencil 2 filler transaction record. However, since the time between the first transaction record satisfying stencil 2 records condition and the current transaction message is 10 hours, exceeding the stencil 2 filler maximum time range of 6 hours, the pattern occurs output value is “0”.
5. Same as card 4, however, this time the stencil 2 filler maximum time condition is met. The pattern occurs value hence is “1”.
6. Same as card 1, however, there is only one transaction record satisfying stencil 1 records condition. The pattern occurs value hence is “0”.
7. Same as card 1, however, there is no transaction record satisfying stencil 2 records condition. The pattern occurs value hence is “0”.
8. Same as card 1, however, there are 6 stencil 1 filler transaction records. The pattern occurs value hence is “0”.
9. Same as card 1, however, there are no stencil 1 filler transaction records. The pattern occurs value hence is “0”.
10. Same as card 1, however, maximum number of allowed filler transactions and maximum time ranges are assumed. The pattern occurs value hence is “1”.

Remarks

- If you are in doubt how patterns compute, use the IBM Safer Payments test function to create various transaction record sequences and check how they are computed.
- All intervals (time ranges, filler numbers) are inclusive, that is, the “to” and “from” values themselves are included in the interval.
- If the “pure” option is enabled for records of a stencil, the records satisfying the conditions may not be interleaved by other records.
- If the record occurrence is set to “1”, the “pure” and “time range” options are not available.
- The number of stencils is not limited.
- The filler time and filler transactions settings are logically “and”-ed. They must both be satisfied for the pattern output attribute to be set to one.
- Occurrence defines how many transactions fitting a stencil’s condition must occur. If there are less transaction messages than the defined number of occurrences directly after another that all fit the conditions, the stencil is not considered applying (and thus the pattern is not considered to match).

back to top

7.5.7.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.7.3 Stencil

A stencil is a combination of a condition that a past transaction record shall satisfy, and a range of “filler” transaction messages (or a “filler time range”) that may occur between two different stencils. All stencils defined must be satisfied by the transaction record sequence in order for the pattern to be matched.

More information on how stencils are computed is found on the Online Help Pattern page.

back to top

7.5.7.3.1 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.7.4 Pattern Attributes

Each pattern profiling creates exactly one new attributes that it feeds its computational results into.

Each of the pattern output attributes is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Profiling generated attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored for this attribute on disk.

back to top

7.5.8 Events

Events track how long ago (in transaction time) a certain occurrence occurred. The occurrence is often a specific (non-monetary) transaction message that is identified by the event's conditions. Each event contains a new numeric type attribute definition that carries the result. The attribute may have any numeric size.

Calendar profiles and events are a very powerful feature in advanced fraud prevention. Whilst the data cache can only support the storage (and thus subsequent access/evaluation by counters) of a limited number of transactions, calendar profiles and events do not have a transaction limit.

Events are used to “remember” things that happened (possibly a long time ago). Examples could be “account opened”, “new card imprinted”, “PIN OK verified”, etc.

An event is a “normal” transaction, however, only the timestamp and event condition attributes are used. The event conditions would typically be implemented by defining (using the MTID meta attribute) an attribute, and defining the condition, for example the MTID value for "PIN verified OK".

Any time a transaction comes in that fires the events condition, the transactions timestamp is stored in the event. The event output attribute reflects the time difference between each new transaction and the last occurrence of the event transaction (for the event transaction itself, this time difference is zero). The smallest unit for time differences is seconds. To make the definition of events easier, minutes, hours, days, weeks, months or years can also be used as definition units. In addition, attributes may use decimals, representing for instance “2.4 days since last address change”.

Remarks

- While typically, event transactions are defined to only “trigger” the event, this need not be the case. An event triggering transaction could also be a full monetary transaction.
- Event transactions are (notwithstanding their event specific effects) just “normal” transactions and are also stored within the transaction data or shown in investigation reports.
- The output value of an event can also be negative in case of transaction messages not always being in strict sequence. If for instance for one cardholder, a posted transaction with a transactiondate of yesterday comes in shortly after an authorization request from a restaurant. If there would be an event defining the last restaurant visit as an event, the posted transaction of yesterday would get the value "-1 days" for the event since with respect to the posted transaction, the last restaurant request occurred -1 days ago. Use mathematical comparison operators in conditions to interpret this value in the way you need it.

back to top

7.5.8.1 Event

The settings for each event are:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comment**
  Used in the model revision to describe the event.
- **Index**
  Index this event is defined along to.
- **Timestamp**
  Attribute used as reference for event computation.
- **Time unit**
  Event time is computed using this time unit.
- **Include current**
  If checked, the event evaluation includes the current transaction. For instance, if this option is not checked, and no event conditions is provided, the event output attribute lists the time since the previous transaction for this index attribute value had arrived.
- **Update during merging**
  Trigger an event or update its timestamp when a record meets the event's conditions after a merging which uses setting `update calendar profiles and events` or after a manual fraud mark.

Memory consumption

Events are defined with reference to an index. Their memory consumption is computed as:

```
4 Bytes * IndexSize
```

both for MDC and DDC. In addition the new numeric attributes memory consumption must be added.

back to top

7.5.8.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.8.3 Event Attributes

Each event profiling creates exactly one new attribute which is filled with the time that has passed since the event last occurred. Occurrence is defined by the conditions.

The event output attribute is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Profiling generated attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored of this attribute on disk.
- **Data type**
  The event profiling output attribute is of numeric data type. Notice that the value has the time unit set as "Time unit" above with the event. The length/decimals settings should cover the value range that you are interested at. Notice that if an event did not occur yet or did occur so long in the past that the value cannot be represented in the universe of the attribute, the value is clipped at the positive maximum of the universe.

back to top

7.5.9 Counters

Counters are similar to calendar profiles, however they evaluate individual transactions back using a referenced index. Thus counters are more flexible than calendar profiles. They can be defined for a "rolling time period" rather than calendar fixed time periods, and past transaction attribute values can be compared to those of the current one (this for example enables defining a counter for "number of transactions at the same ATM within the past 2 hours"). Also, there are more complex evaluation methods available than in calendar profiles. The disadvantage of counters in comparison with profiles is, that in particular with long-term evaluations, the large amount of transactions that require evaluation can make counters perform significantly more slowly than profiles.

Counter results are stored in one or more new attributes defined within the counter. Counters themselves require a sequence but apart from this and the new attribute have no memory consumption.

back to top

7.5.9.1 Counter

A counter assembles a set of past transaction records that satisfy a time range and a maximum number of evaluated and matching records criterion, as well as a set of conditions, alongside an index sequence. As an option, the current transaction is included in this set, if the "include current" checkbox is checked.

Example

Assuming the following sequence of transactions:

#	Timestamp	Amount	Country	Merchant Category
1	2010-01-10 14:00:00	114.13	US	5571
2	2010-01-16 12:00:00	83.03	US	5411
3	2010-01-18 08:00:00	200.00	US	6011
4	2010-01-18 09:00:00	400.00	MY	5813
5	2010-01-18 09:01:00	400.00	MY	5813
6	2010-01-18 09:02:00	400.00	MY	5813
7	2010-01-18 15:00:00	133.00	US	5812
8	2010-01-18 19:00:00	300.00	US	5933
9	2010-01-22 11:00:00	1018.19	MO	5541
10	2010-02-08 12:00:00	100.00	US	5973
11	2010-02-08 12:15:00	300.00	US	5973

Here, the last row #11 represents the current transaction. The following counter definition examples would imply the following set of transactions:

- **A**
  Include current: no
  Max evaluated records: 100
  Max matching records: 100
  Time range: 21 to 0 days
  Evaluation Conditions: none
  => past transaction records 7 to 10 counted
- **B**
  Include current: no
  Max evaluated records: 100
  Max matching records: 100
  Time range: 4 to 0 weeks
  Evaluation Conditions: Country equal to US
  => past transaction records 2, 3, 7, 8, and 10 counted
- **C**
  Include current: no
  Max evaluated records: 3
  Max matching records: 3
  Time range: 4 to 0 weeks
  Evaluation Conditions: Country equal to US
  => past transaction records 8 and 10 counted
- **D**
  Include current: yes
  Max evaluated records: 100
  Max matching records: 100
  Time range: 4 to 0 weeks
  Evaluation Conditions: Country equal to US
  => past transaction records 2, 3, 7, 8, 10, and 11 counted
- **E**
  Include current: yes
  Max evaluated records: 100
  Max matching records: 100

Time range: 1 to 0 weeks
Evaluation Conditions: none
=> past transaction records 10 and 11 counted

Based on this set of past (and potentially the current) transactions, a number of output attributes is derived. For each output attribute, the "computation" setting defines how this attribute shall be computed from the transaction set. For details on computation methods, open the online help page of the output attribute.

Remarks

Please notice, that the time range definition is inclusive, i.e. time range 3 to 0 hours means 0 <= x <= 3.

back to top

7.5.9.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.5.9.3 Counter Attributes

Each counter profiling creates one or more new attributes that it feeds its computational results into. Notice that if you use a mandator structure, the rules of this decision logic may use all attributes defined in champion mandator revisions above it in the structure. Each of the counter output attributes is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain. Notice that the attribute names do not need to correspond to the variable names of data delivered to IBM Safer Payments; you define the relation between IBM Safer Payments attributes and variable names in "Mappings".
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be in the MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored for this attribute on disk.
- **Data type**
  Computations (see below) either deliver numeric or text output (only if computation is "most occurrence value" the output attribute of a counter may be "text").
- **Formatted as**
  The formatting options are for display of values on the IBM Safer Payments pages (for examples in queries or case investigation). Choices differ by data type:
  - **Amount**
    Using digit group and decimal separators as defined for each user's preferences (e.g. "12,345.67") for numeric attributes only.
  - **Decimals**
    Using decimal separators as defined for each user's preferences (e.g. "12345.67") for numeric attributes only. This option does not use digit group separators.
  - **ID**
    Using digit group separators as defined for each user's preferences (e.g. "123,456,789") for numeric attributes only.
  - **PAN**
    Using dashed quadruple format typically used for primary account numbers as embossed on cards (e.g. "1234-1243-1243-1243") for numeric and text attributes.
  - **No formatting**
    Shows data with no formatter applied.
- **Length/decimals**
  Byte length of internal storage for numeric output attribute values, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.
- **Unit**
  Displayed with numeric values of this attribute.
- **Computation**

This setting determines how the output attribute is computed from the set of transaction records selected by this counter:

- **Simple counting methods**
  The simple counting methods include:
  - **Frequency**
    Number of transaction records selected.
  - **Total amount**
    Total amount of all transaction records selected.
  - **Average amount**
    Average amount of all transaction records selected.
  - **Maximum amount**
    Maximum amount of all transaction records selected.
- **Multiple occurrences**
  The overly frequent occurrence of certain values for an attribute can be a strong indicator of fraudulent behavior. IBM Safer Payments provides different counting methods for this indicator:
  - **Most occurrence frequency**
    This counting method outputs the number of selected transaction records that have the same value as the reference attribute (defined above). If for instance this counter is used on MerchantID with the index/sequence attribute PAN, this counting method returns how many transactions the cardholder of the current transactions had at most with the same MerchantID.
  - **Ratio of most occurrence**
    This is the normalized version of the previous. It is computed as the most occurrence frequency value divided by the total number of transaction records that satisfy the counter conditions (frequency). If no transaction satisfies the counter conditions, this value is zero. If the value occurs no more than once, this value is also zero. Notice that this value is always in the interval [0; 1]. You should therefore define the respective counter output attribute with decimals.
  - **Most occurrence value**
    Delivers the value that occurred most. Can be used with text and numeric type reference attributes. The respective counter output attribute must have the same type as the referenced attribute. If the value occurs no more than once, this computation delivers zero for numeric type attributes and an empty text for text type attributes. If more than one value occurs with the maximum same frequency, the first value is outputted. Notice that with this computation method, data type and length of the output attribute gets taken from the reference attribute and may not be changed. If categories are defined for the reference attribute, they will be assigned to the output attribute of the counter automatically. In this case the categories are shown at the bottom of the page but are not editable.

  Examples: Assume the following lists of values of the reference attribute with transaction records that satisfy the counter conditions:

Values of reference attribute	Most occurrence frequency	Ratio of most occurrence
{10; 20; 20; 30; 30; 30; 40; 50}	3	3/8=0.375
{}	0	0.000
{10}	1	0.000
{10; 10; 10}	3	1.000
{10; 10; 10; 10; 20; 20; 20; 20}	4	4/8=0.500
{10; 20; 30; 40; 50; 60; 70; 80}	1	0.000
{10; 20; 20; 30; 30; 30; 40; 10}	3	3/8=0.375

- **Distinct values**
  Some fraud patterns can be identified by analysing the number or ratio of distinct values of the reference attribute in the past transaction record sequence. IBM Safer Payments provides different counting methods for this indicator:
  - **Number of distinct values**
    This counting method outputs the number of values of the reference attribute of past transactions that are different from each other.
  - **Ratio of distinct values**
    This is the normalised version of the previous counting method. It is "0" for all values being the same (if more than one value is counted), and "1" for all values being different from each other.

  Examples: Assume the following lists of values of the reference attribute with transaction records that satisfy the counter conditions:

Values of reference attribute	Number of distinct values	Ratio of distinct values
{10; 20; 20; 30; 30; 30; 40; 50}	5	4/7=0.571
{}	0	1.000
{10}	1	1.000
{10; 10; 10}	1	0.000
{10; 20; 30}	3	2/2=1.000
{10; 10; 10; 10; 20; 20; 20; 20}	2	1/7=0.143
{10; 20; 30; 40; 50; 60; 70; 80}	8	7/7=1.000
{10; 20; 20; 30; 30; 30; 40; 10}	4	3/7=0.429

- **Multiple values**
  Some fraud patterns can be identified by analysing the number or ratio of values of an attribute in the past transaction sequence that occur more than once. IBM Safer Payments provides different counting methods for this indicator:

- **Number of multiple values**
  This counting method outputs the number of values of the reference attribute of past transactions that occur more than once.
- **Ratio of multiple values**
  This is the normalised version of the previous counting method.

Examples: Assume the following lists of values of the reference attribute with transaction records that satisfy the counter conditions:

Values of reference attribute	Number of multiple values	Ratio of multiple values
{10; 20; 20; 30; 30; 30; 40; 50}	2	2/8=0.250
{}	0	0.000
{10}	0	0.000
{10; 10; 10}	1	1/3=0.333
{10; 20; 30}	0	0.000
{10; 10; 10; 10; 20; 20; 20; 20}	2	2/8=0.250
{10; 20; 30; 40; 50; 60; 70; 80}	0	0.000
{10; 20; 20; 30; 30; 30; 40; 10}	3	3/8=0.375

- **Distinct IP B/C Nets**
  This is a variant of "number of distinct values" designed to count the number of different B/C IP nets used by a customer. For this, an attribute holding the IP address must be specified as reference attribute. The format of the IP address must be the "dotted" format, such as "129.44.1.8". The subnet mask applied is "255.255.255.0" for distinct C class nets and "255.255.0.0" for distinct B class nets. That is, IP addresses "129.44.1.2", "129.44.1.255", "129.44.1.0", and "129.44.1.144" would all be considered to be with the same C class net as "129.44.1.8"; and thus not counted as distinct nets. Notice that the IP address holding attribute must be of "text" type and have a minimum length of 15 ("xxx.xxx.xxx.xxx").

back to top

7.5.10 Collusions

Business background

Collusion is the process where card information is copied from transactions made at one or more terminals at the merchant site ("common point of purchase", aka CPP or "point of compromise", aka POC), over a defined period of time, and later used to create fraudulent transactions.

A traditional way of fighting this type of fraud is to replace all cards that have been used at the POC during the affected period. This is a costly process involving customer disruption. It also only works after a point of compromise is detected and verified.

IBM Safer Payments provides a much better way to deal with this type of fraud. Using collusion processing, IBM Safer Payments constantly analyses past transactions for indicators that a set of cards has been compromised and is now used for fraudulent transactions. Special investigation cases are generated for potential points of compromises. These investigation cases provide the possibility to protect compromised cards even before they are used for fraudulent transactions. In addition collusions not only detect potential points of compromise but als provide further information such as a skimming time range.

In this way, collusion fraud can be prevented even before a POC is known, and POCs themselves can be found.

Collusion processing can be triggered by rules, if the respective conditions are met, or when mergings are executed. With that it is possible to execute a collusion processing automatically in all cases where cards seem to be used fraudulently after being compromised in order to protect other cards before collusion fraud takes place. Collusions are triggered within the real-time processing of incoming messages but are executed in parallel to the real-time execution of messages. Using this technology collusion processings will not influence the computation time of incoming transactions.

In addition to the automatic execution of collusions it is possible to run collusions manually for a predefined data selection.

For reasons of generality, what in the standard situation described above is the "card" or the "account" is referred to as "first party", what is the "merchant" or the "terminal" is the counterparty".

Analytical process

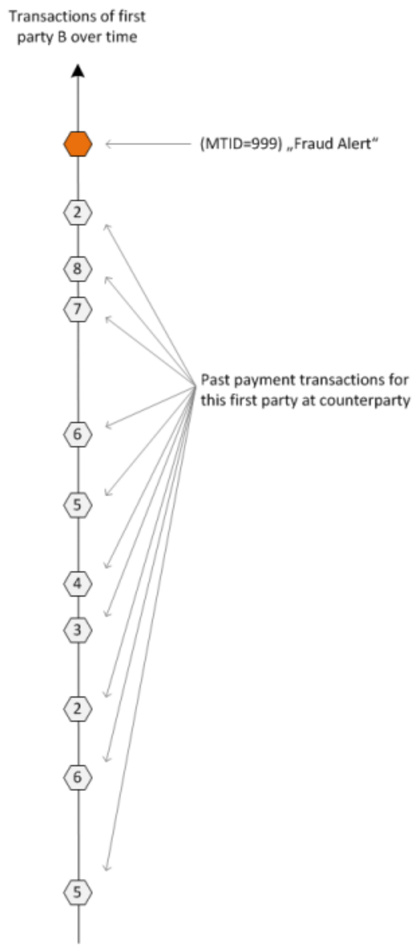
The analytical process is as follows:

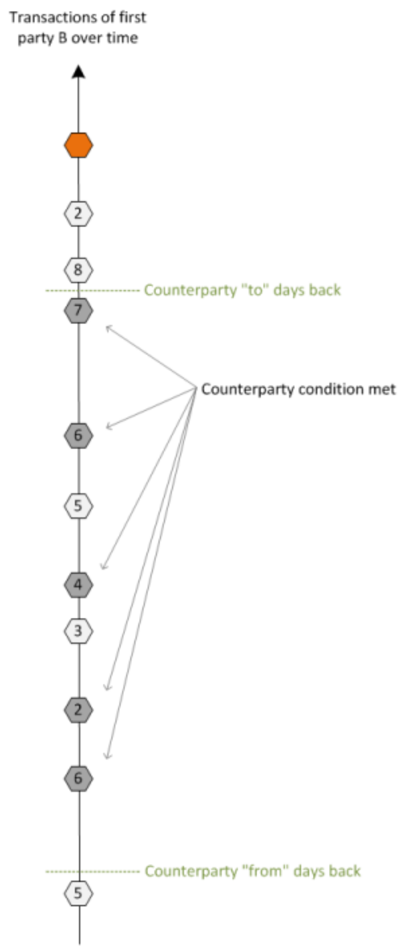
1. Analyse the transaction record history for the first party. Collect all transaction records of this history that satisfy the “counterparty time criterion” and “counterparty conditions”. The counterparty time criterion is a time interval (from/to) relative to the value of the meta attribute 'Timestamp'.
2. For the resulting set of past transaction records, create a set of counterparties involved with these transaction records.
3. For each counterparty, select what other first parties had transaction records with the counterparty, with the transaction records satisfying the “connivance time criterion” (relative to the transaction message timestamp found in step 2. above) and the “connivance conditions”.
4. Count how many of these first parties satisfy the compromised criteria within the compromised time range (with respect to the reference timestamp) after they made the first transaction at the counterparty within the connivance time period; and do not satisfy the compromised criteria before. If no compromised conditions are defined the compromised criteria includes fraudulent transaction. If additional compromised conditions are defined, first parties are counted as affected first parties if either "fraud-criterion" or the compromised conditions are satisfied within the compromised time range.
5. For each counterparty where more first parties than defined as "Threshold first parties" are counted, generate the defined “point of

compromise suspicion case” (as defined as case class).

Example

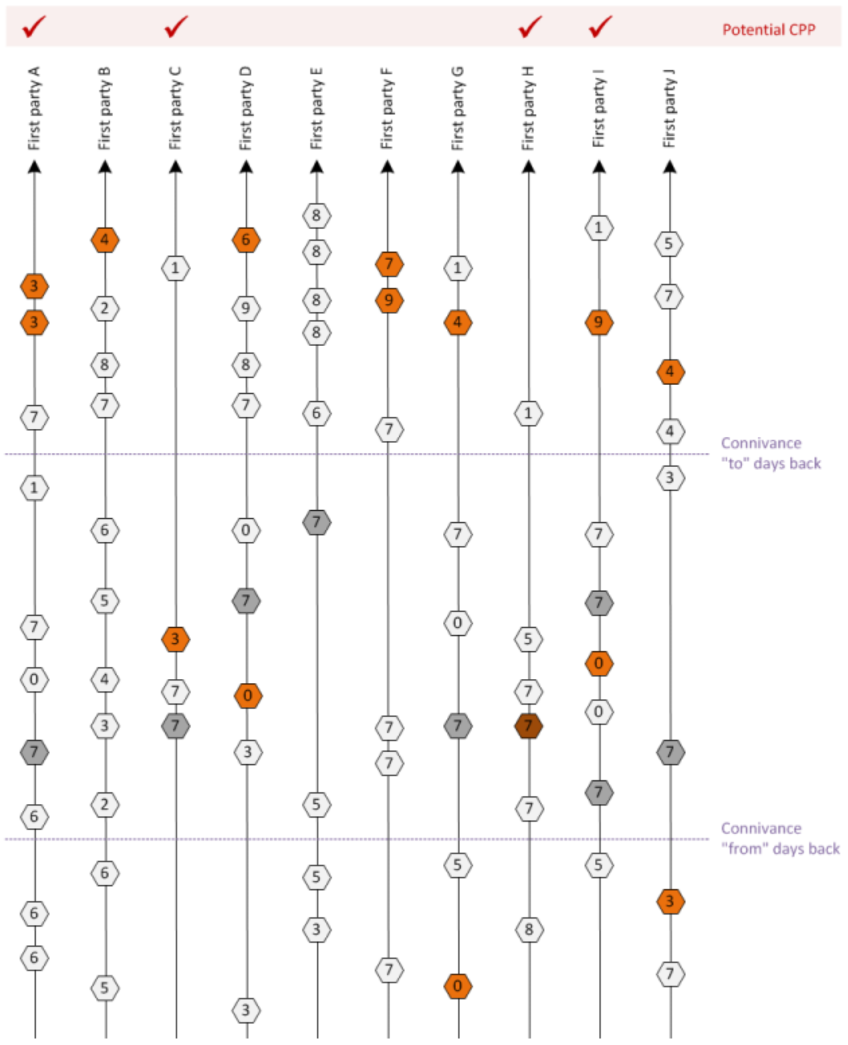
Assuming the conditions of a rule that indicates fraudulent behaviour induced by a POC are satisfied, the respective rule triggers a collusion. This transaction is marked red and the previous transactions are shown with their counterparty value in the figure below left:


Transactions of first party B over time
(MTID=999) „Fraud Alert“
2
8
7
Past payment transactions for this first party at counterparty
6
5
4
3
2
6
5


Transactions of first party B over time
2
8
Counterparty "to" days back
7
Counterparty condition met
6
5
4
3
2
6
Counterparty "from" days back
5

Now the sequence of past transaction records is analysed: the "counterparty time range" identifies the sub-sequence (within dotted green lines) in which merchants are considered that had transaction records with first party B. In the figure above right, these were the counterparties 2, 4, 6, and 7 (marked dark grey). Counterparties 3 and 5 do not fulfil the counterparty conditions.

Next the past transaction record sequences of all four counterparties are analysed as potential POCs. The next figure shows the POC analysis for the first counterparty 7:



The figure shows the first party transaction record histories (vertical lines) for each first party that had transactions with the counterparty 7 within the connivance period. Fraudulent transaction records are marked in red, transaction records involving counterparty 7 that match the connivance conditions are marked dark grey. The counterparty values are printed in the hexagons.

The first party transaction record sequences are evaluated:

- First party A: The first party had one transaction record at counterparty 7 within the connivance period and fraud within the compromised time range after the first transaction record at counterparty 7 and no fraud before the first transaction record at counterparty 7. It is thus counted as potential POC induced fraud.
- First party B: This is the first party whose transaction message is currently/originally evaluated. Since the first transaction record at counterparty 7 occurred after the connivance period, this first party is not considered further.
- First party C: The first party has a transaction record within the connivance period at counterparty 7, no fraud before the first transaction record at counterparty 7, and at least one fraud transaction record within the compromised time range after the first transaction record at counterparty 7. It is thus counted as potential POC induced fraud.
- First party D: The first party has a transaction record within the connivance period at counterparty 7, but fraud occurred before the first transaction record at counterparty 7. This first party is not considered further.
- First party E: The first party has a transaction record within the connivance period at counterparty 7, but no fraud occurred. This first party is not considered further since no additional compromised conditions are defined.
- First party F: The "connivance conditions" are not satisfied for the transaction record at counterparty 7. This first party is not considered further.
- First party G: Same as first party D. This first party is not considered further.
- First party H: The first transaction record at the counterparty that satisfies the connivance conditions is fraudulent. It is thus counted as potential POC induced fraud.
- First party I: Fraud occurred within the compromised time range after the first transaction record with the counterparty. It is thus counted as potential POC induced fraud.
- First party J: Even though the first fraud occurred after the first transaction record at the counterparty and within the compromised time range, this first counterparty transaction record was outside the connivance period. This first party is not considered further.

The same analysis is performed on all four counterparties. The example results of this are:

Counterparty	2	4	6	7
First parties that indicate potential POC	14	33	1	4
All first parties that fit connivance criteria	22	193	5	9
Ratio	38.89%	17.10%	1.85%	44.44%

This concludes the analytical part of collusions. Based on the results:

- POC investigation cases are generated through the generation of alarms, and
- the indicators computed are displayed within the newly created cases or updated if the generated alarm is aggregated with an already existing case for the found potential point of compromise.

A POC alarm is generated for each counterparty where the result is higher than pre-defined threshold:

- Threshold first parties
  Absolute number of first parties that indicate potential POC.

In addition to the detected potential POC the following information are provided within the generated cases:

- number of affected first parties
- number of generated alarms
- skimming time range

Remarks

- Collusion alarms also contain a list of potentially affected first parties. These are displayed on the case investigation page in a separate section and can be drilled down for their past transaction record sequence.
- Besides the time ranges mentioned above there are also limits available for first parties and counterparties which are included in the collusion processings. This allows to control the computation time of collusions.
- Manually executed collusions do not generate cases. Instead a summary is displayed where all detected potential POCs are displayed together with further information such as number of affected first parties and the respective skimming time range. These information can be used for further investigations.

back to top

7.5.10.1 Collusion

Each collusion searches within past transactions whether there is an accumulation of fraud with accounts/cards that could indicate that a common point of purchase has been used to extract account/card information. Refer to the online help function for collusions above for details on the computation. Each collusion comprises a number of definitions:

In order to include a collusion within the real-time processing of transactions, the respective checkbox has to be enabled for a rule or a merging.

To simulate a collusion processing press the green button on top of the form. This will start a simulation of the respective collusion using the data selection configured on the page "Simulation". Results of manually executed collusions are provided in a table on the bottom of this page. It is recommended to simulate collusions before they are taken into production to avoid a large number of generated alarms or unexpected results. Manually executed collusions can be aborted at any time during computation by clicking the respective red button.

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **First party (index)**
  Index of the attribute (must have a sequence) that shall be used as first party. This would typically be the attribute that identifies the card or the account.
- **Threshold first parties**
  Threshold for the absolute number of cards that indicate a potential POC. If this threshold is reached for a counter party, a collusion alarm is generated.
- **Counterparty (index)**
  Index of the attribute (must have a sequence) that identifies the counterparties to be evaluated. This would typically be the attribute that identifies the merchant or the terminal.
- **Counterparty time range**
  Time interval, relative to the value of the meta attribute 'Timestamp', in which counterparties that had transactions with the first party are considered. Only counterparties are considered as points of compromise that had transactions with the first party during this time period and that also satisfy the "counterparty conditions" (below). Please notice, that the time range definition is inclusive, i.e. time range 3 to 0 hours means 0 <= x <= 3. Please also notice that the sequence timestamp attribute of the first party index is used for this evaluation.
- **Max counterparties**
  The maximum number of counterparties that will be evaluated.
- **Connivance time range**
  Time before and after the transaction at the counterparty, in which other first parties are considered that had transactions with a counterparty and that also satisfy the "connivance conditions" (below). Please notice, that the time range definition is inclusive, i.e. time range 3 to 0 hours means 0 <= x <= 3. Please also notice that the sequence timestamp attribute of the counterparty index is used for this evaluation.
- **Max first parties**
  The maximum number of first parties, that will be evaluated for each counterparty.
- **Compromised time range**

Time interval, relative to the current reference timestamp, in which the compromised criterion (below) has to be satisified. The reference timestamp is the most recent timestamp that is available for the meta attribute 'Timestamp'. Please note that in case no compromised conditions are defined, only fraudulent transactions are considered as indicators for a compromised account.

- **Generate alarms**
  An alarm is generated for each counterparty that is considered a potential point of compromise:
  - **Case class**
    Id of case class that the generated alarm(s) shall belong to. Please note that only case classes of type "collusion" are selectable.
  - **Case score**
    Value of the meta attribute "case score" which determines the importance of the generated alarm(s).

For help on the sections, refer to their individual online help pages.

Notice that the online help page for collusions (from the top section) describes the collusion evaluation process in detail.

back to top

7.5.10.2 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

7.6 Formulas

Formulas allow the use of mathematical expressions to generate a new attribute from existing ones. While mathematical expressions can also be used with conditions, using a formula to generate a new attribute can be more efficient if this expression is used many times in the model revision. The attribute may have any numeric size.

Note that rule conclusions that write a formula expression to an output attribute are similar to Model/Formula.

back to top

7.6.1 Formula

For details on formula definition, rest the mouse pointer over the respective field to show tooltip style explanations.

There are six operations each of which has two operands:

- (x + y)
- (x - y)
- (x * y)
- (x / y)
- geoDistanceKm(pos(latitudeA; longitudeA); pos(latitudeB; longitudeB))
- geoDistanceMiles(pos(latitudeA; longitudeA); pos(latitudeB; longitudeB))

The operands x, y, latitudeA, longitudeA, latitudeB and longitudeB may either be:

- a numeric expression (decimal character is period, no spaces in numbers, may have leading minus sign)
- an attribute of the same transaction (format: [attribute name]) or
- a math condition

In addition to all operations mentioned above, you can also make Python function calls in your formulas. To get the list of available Python functions you need to type "py" in the expression field. For more information about Python code execution refer to the respective online help page.

Memory consumption of a formula is only the memory consumption of the formula output attribute. There is no overhead.

pos is a placeholder formula to encode 2 values for geo distance processing. It cannot be used without geoDistanceKm or geoDistanceMiles

geoDistanceKm / geoDistanceMiles computes the distance between two geographical coordinates.

Following steps should be performed to achieve useful distance calculation results:

- Define latitude/longitude masterdata for the respective index attribute.
- Define message type, and corresponding mapping to import geo locations
- Load geographical coordinates using a batch load job. The csv file should contain following attributes: MTID, zip, latitude, longitude)
- Define three precedents to capture the data of the previous transaction: previousLatitude, previousLongitude, previousTimestamp

- Define formula to compute distance: geoDistanceKm(pos(#previousLatitude; #previousLongitude); pos(#latitude; #longitude))
- Define formula to compute time difference: sub(#trxDateTime; #previousTimestamp)
- Define rule(s), e.g. (zipDistanceKm ≥ 75) ; (zipTimeDifference ≤ 3600); (trxType != eCommerce)

back to top

7.6.2 Formula Attributes

Each formula creates exactly one new attribute which is filled with the result of the calculation.

The formula output attribute is specified by a set of definitions that are made on this form:

- **Name**
  The name is used in all IBM Safer Payments forms and should be chosen from a business domain.
- **Comments**
  Comments are only for documentational purposes. It is advisable to comment the attributes extensively, so the decision logic remains easy to understand.
- **Storage type**
  Attributes that you need in real-time (for counters and mergings) or for analysis and rule generation should be stored in MDC and DDC. Attributes that you only need for investigation and queries should only be stored in the DDC. Attributes that are only used for the evaluation of the current transaction and for which you do not need any history do not need to be stored at all. Notice that your storage options determine how much main and disk memory IBM Safer Payments consumes (number of records times length/characters). You find the memory totals for this model revision in "General".
- **MDC records**
  Number of records that should be stored of this attribute in main memory. Because data in main memory is not persistent, the MDC is primed from the DDC when IBM Safer Payments starts up. This implies that the DDC size (i.e. the number of records stored) must always be greater than or equal to the MDC size.
- **DDC records**
  Number of records that should be stored of this attribute on disk.
- **Data type**
  The data type of a formula output attribute is numeric.
- **Length/decimals**
  Byte length of internal storage, ranging from 1 to 8, and decimals ranging from 0 to 6. The value range that the resulting attribute can represent is computed live in the browser and displayed on the right.

back to top

7.7 Model Components

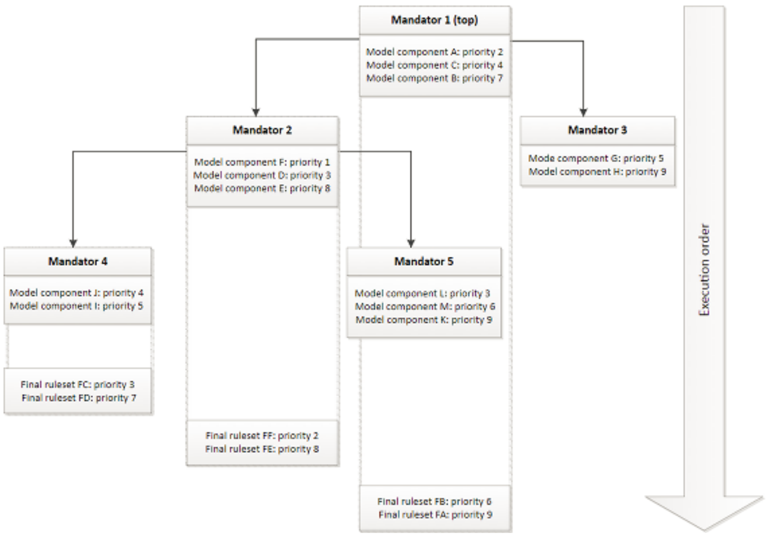
This section defines modeling capabilities of IBM Safer Payments. Safer Payments provides the possibility to combine multiple modeling types to combat fraud effectively. While some of them (rulesets) can be created and trained within Safer Payments itself, other model types can be imported to Safer Payments in PMML (Predictive Model Markup Language) format. The imported PMML models can be evaluated as a part of the overall model alongside model components created within IBM Safer Payments. The following model component types are supported:

- Ruleset / Scorecard
- Decision tree
- Neural network
- Random forest

General remarks

- With all the model components of this model revision, a model component with higher priority is computed after a model component with lower priority. This is because later model components can overwrite output of earlier model components.
- First the model components of the "highest" mandator in the hierarchy are computed, all model components of the current mandator are computed last. This is equivalent to an automatically higher priority of the model components of the current model revision, since model components in the current model revision can overwrite any decision a model component of a higher mandator's champion model revision has made. This implies that the model component priorities only determine the computation lineup of model components within a model revision.

The following diagram exemplarily shows an order of execution of model components and final rulesets.


Mandator 1 (top)
Model component A: priority 2
Model component C: priority 4
Model component B: priority 7
Mandator 2
Model component F: priority 1
Model component D: priority 3
Model component E: priority 8
Mandator 3
Mode component G: priority 5
Model component H: priority 9
Mandator 4
Model component J: priority 4
Model component I: priority 5
Mandator 5
Model component L: priority 3
Model component M: priority 6
Model component K: priority 9
Final ruleset FC: priority 3
Final ruleset FD: priority 7
Final ruleset FF: priority 2
Final ruleset FE: priority 8
Final ruleset FB: priority 6
Final ruleset FA: priority 9
Execution order

The model components and final rulesets are executed in the following order (Assumption: current mandator = mandator 4):

1. All model components of the top mandator (mandator 1) according to their priorities (A -> C -> B)
2. All model components of the head mandator (mandator 2) according to their priorities (F -> D -> E)
3. All model components of the current mandator according to their priorities (J -> I)
4. All final rulesets of the current mandator according to their priorities (FC -> FD)
5. All final rulesets of the head mandator (mandator 2) according to their priorities (FF -> FE)
6. All final rulesets of the top mandator (mandator 1) according to their priorities (FB -> FA)

PMML specific remarks

- IBM Safer Payments supports import and execution of a subset of model types and options defined in PMML version 4.3.
- Refer to the section help data field mapping to see how data field mappings can be defined.
- Refer to the section help output field mapping to see how output field mappings can be defined.
- Refer to the section help transformation field mapping to see how transformation field mappings can be defined.
- Before uploading large PMML model files, you might need to adjust the maximum post size setting on the system configuration page. Uploads which exceed the maximum allowed post size will be rejected.
- Note that wildcard characters encountered in PMML predicates will be interpreted according to their role in IBM Safer Payment's expressions.

PMML data types

Below you can find all supported PMML data types and their counterparts in IBM Safer Payments to which a valid mapping can be defined.

PMML data type	SP data type
string	text, IPv4, hexadecimal, numeric, timestamp
integer	numeric
float	numeric
double	numeric
boolean	boolean
dateTime	timestamp

PMML transformations

PMML transformations allow to derive new data fields from input data by applying various functions on them. IBM Safer Payments supports a subset of these transformations defined in PMML 4.3 standard. The table below provides an overview of the supported transformations per PMML model type.

	PMML decision tree	PMML random forest	PMML neural network
Constant	Transformation dictionary and outputs with feature "transformedValue"	Transformation dictionary and outputs with feature "transformedValue"	-
FieldRef	Transformation dictionary and outputs with feature "transformedValue"	Transformation dictionary and outputs with feature "transformedValue"	Input layer
Apply	Transformation dictionary and outputs with feature "transformedValue"	Transformation dictionary and outputs with feature "transformedValue"	-
NormDiscrete	-	-	Input layer
NormContinuous	-	-	-
Discretize	-	-	-
MapValues	-	-	-
TextIndex	-	-	-
Aggregate	-	-	-
Lag	-	-	-

The following built-in functions are supported in "Apply" transformations: **+**, **-**, *****, **/**, **log10**, **ln**, **sqrt**, **abs**, **exp**, **pow**, **threshold**, **floor**, **ceil**, **round**, **min**, **max**, **sum**, **avg**, **product**.

back to top

7.7.1 Ruleset / Scorecard

Rulesets are a structuring aid for rules. Grouping rules into sets has a number of advantages:

- Often the representation of a certain fraud type or region specific fraud cannot be done in one rule. Grouping them into a ruleset keeps them at one place for easier reference and maintenance.
- Rulesets can be enabled and disabled entirely. This allows for a set of rules to always be enabled and disabled at once.
- Rulesets can have additional conditions. Thus if there is a number of rules that shares some conditions (such as region, merchant type, or similar), this condition must only be formulated once as the ruleset condition, keeping the actual rules conditions smaller and easier to maintain.
- Because each ruleset can be given a priority, grouping rules in sets can make it easier to create and maintain a priority scheme for the entire rules of the model revision.

Remarks

- Notice that the individual priorities of single rules in ruleset are only relevant for the order of execution within a ruleset.
- Also notice that the conditions that you define for a ruleset will be applied as if they would be defined the same for all individual rules of the set.
- In some applications, there are rules that *need* to be computed last (i.e. with highest priority). If you need this, enable "Final rulesets" within the mandator administration settings for the respective mandator(s). You find more information on final rulesets on their online help page.

back to top

7.7.1.1 Rules

Rules are the core of the model revision as they combine values of input attributes and attributes generated by the profiling methods listed before in this section, to identify fraud patterns. Rules can also have any number of conclusions that set or modify output attributes.

Rules can be entered manually, generated assisted/automatically or be imported via a PMML scorecard.

- To manually enter a new rule, click on the [New rule] icon. To edit an existing rule, click on the respective rule row. Extended rule options are available from a context menu by clicking right on the respective row. You may also select multiple rules using the [Ctrl] key with mouse clicks for individual selection or the [Shift] key with mouse clicks for the selection of an interval.
- Information on assisted or automatic rule generation can be found on the Automatic and Assisted Rule Generation online help page.
- To import rules via a PMML scorecard, click on the [Import PMML scorecard] icon or drop the file directly on the icon. Further information on PMML scorecard import can be found on the PMML online help page.

Remarks

With all the rules of this ruleset, a rule with higher priority is computed after a rule with lower priority. This is because later rules can overwrite conclusions of earlier rules.

Rules require no memory in MDC or DDC.

back to top

7.7.1.1.1 Rule

The definition of a rule involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  If checked, this rule is active.
- **Priority**
  Enter any value between 0 and 10,000 for the rule priority, the higher this value is, the LATER the rule is computed within this ruleset (later computation means it can overwrite computations by earlier computed rules).
- **Name**
  Name of this rule.
- **Comment**
  Comments are informational only, however it is highly recommended that you comment each item in IBM Safer Payments fully as these comments are used in various places.
- **Performance report**
  Includes this rule for performance report analysis performed using group by queries. If you enable this setting, the rule will keep a record for which transaction it was applied. Thus, this requires additional memory in MDC or DDC.
- **Collusions**
  Selected collusions are triggered if the rule conditions are met. For further information about collusions please refer to the respective online help page.
- **Conditions**
  The conclusions of a rule are executed when all defined conditions are satisfied. If no condition is defined, the conclusions of a rule will be applied to all transaction messages.

- **Conclusions**
  The conclusions of a rule are executed when all defined conditions are satisfied.
- **Actions**
  The actions of a rule are executed when all defined conditions are satisfied.

If a rule query is defined, you can execute it on a specific rule by selecting "Execute rule query" from the context menu or the form. A rule query delivers all records, that are hit by the selected rule. This function requires an active and valid simulation. Further information on simulation queries can be found here.

Selecting "Execute Rule Analysis" from the form starts a single rule analysis in order to get statistics about this specific rule. Further information on rule analysis can be found here.

Information about the rule report can be found here.

back to top

7.7.1.1.2 Rule Report

Important: To start a rule report a rule report query has to be defined and the simulation of the meta attribute intercept has to be enabled in the modelling section. Beside that the executing user needs to have simulation memory for the mandator of this revision.

The rule report provides information about the performance of a rule. There are two options to start the rule report:

- **With context**
  The complete decision model is considered. To start the rule report with context it is required to have a valid simulation already.The summary takes only those records into account where the rule, the report is generated for, fired. However the relative amount and share against the entire Simulation are shown in the summary as well. The absolute numbers for the entire Simulation can be found in the report header. Note that this option is only available if the ruleset and the rule are enabled.
- **Without context**
  Only the selected rule is simulated. In case you start a report "without context", all your previous simulation and analysis results will be invalid. Notice that results will be lost as soon as you close the form. The executing user needs to have memory assigned to his user account for the mandator of this revision.

The report consist of three parts:

- **Report Settings**
  The basic information about the report is listed here. Beside that the definition of the rule set and the rule itself is shown.
- **Summary**
  This section summarizes the number of generated alarms dissected by the return value (meta-attribute intercept). The summary is computed using the simulation the report based on, not the original transaction data.
- **Record list**
  The record list shows the records where the rule fired. It is defined as a rule report query. The record list is computed using the simulation the report based on, not the original transaction data.

back to top

7.7.1.1.3 PMML Scorecard Import

PMML (short for Predictive Model Markup Language) is a XML-based file format used to describe and exchange models produced by data mining and machine learning algorithms. One of the supported models is the scorecard model which you can use to import rules into IBM Safer Payments.

XML schema

The PMML scorecard model has the following xml schema:

```
<xs:element name="Scorecard">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
   <xs:element ref="MiningSchema" />
   <xs:element ref="Output" minOccurs="0" />
   <xs:element ref="ModelStats" minOccurs="0" />
   <xs:element ref="ModelExplanation" minOccurs="0"/>
   <xs:element ref="Targets" minOccurs="0" />
   <xs:element ref="LocalTransformations" minOccurs="0" />
   <xs:element ref="Characteristics" />
   <xs:element ref="ModelVerification" minOccurs="0" />
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="modelName" type="xs:string" />
  <xs:attribute name="functionName" type="MINING-FUNCTION" use="required" />
  <xs:attribute name="algorithmName" type="xs:string" />
  <xs:attribute name="initialScore" type="NUMBER" default="0" />
  <xs:attribute name="useReasonCodes" type="xs:boolean" default="true" />
  <xs:attribute name="reasonCodeAlgorithm" default="pointsBelow">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="pointsAbove" />
      <xs:enumeration value="pointsBelow" />
     </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="baselineScore" type="NUMBER"/>
  <xs:attribute name="baselineMethod" default="other">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="max" />
     <xs:enumeration value="min" />
     <xs:enumeration value="mean" />
     <xs:enumeration value="neutral" />
     <xs:enumeration value="other" />
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="isScorable" type="xs:boolean" default="true"/>
 </xs:complexType>
</xs:element>

<xs:element name="Characteristics">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
   <xs:element ref="Characteristic" maxOccurs="unbounded" />
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

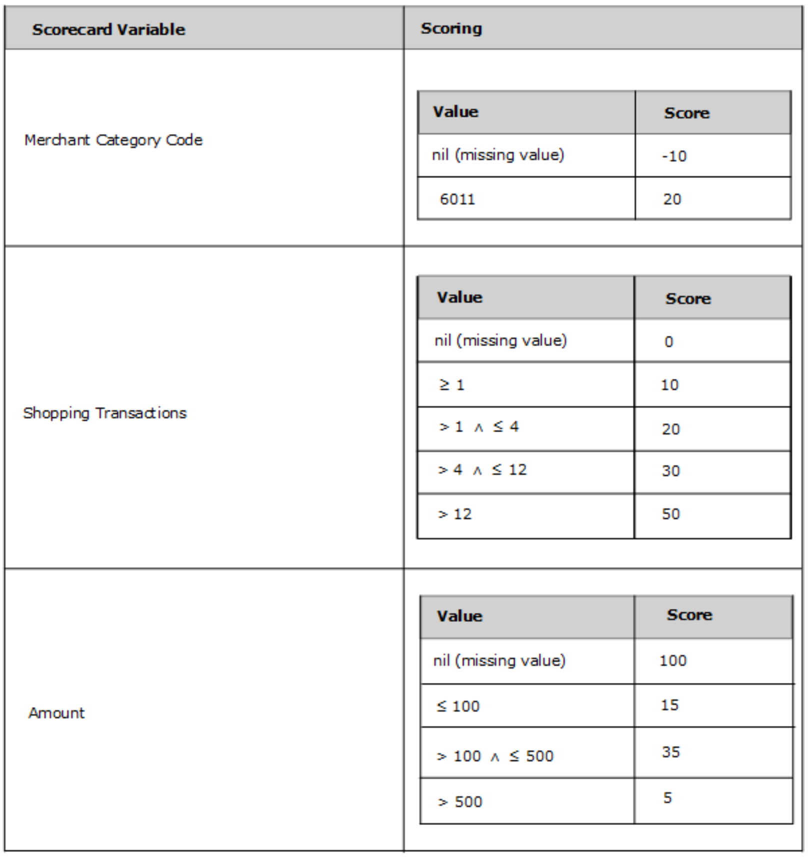
To import rules into IBM Safer Payments from a PMML scorecard you have to upload a valid XML file, corresponding to the schema above, via a browser. To that purpose you can click on the [Import PMML scorecard] icon in the ruleset form or drop the file directly on the icon.

Example

An example for a valid XML file would be following scorecard definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<PMML version="4.1" xmlns="http://www.dmg.org/PMML-4_1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Header copyright="www.dmg.org" description="Sample scorecard">
    <Timestamp>2010-11-10T08:17:10.8</Timestamp>
  </Header>
  <DataDictionary>
    <DataField name="Merchant Category Code" dataType="string" optype="cotegorical"/>
    <DataField name="Shopping Transactions" dataType="integer" optype="continuous"/>
    <DataField name="Amount" dataType="double" optype="continuous"/>
  </DataDictionary>
  <Scorecard modelName="SampleScorecard" functionName="regression"
   useReasonCodes="true" reasonCodeAlgorithm="pointsBelow" initialScore="0" baselineMethod="other">
    <MiningSchema>
      <MiningField name="Merchant Category Code" usageType="active" invalidValueTreatment="asMissing"/>
      <MiningField name="Shopping Transactions" usageType="active" invalidValueTreatment="asMissing"/>
      <MiningField name="Amount" usageType="active" invalidValueTreatment="asMissing"/>
     </MiningSchema>
    <Output>
     <OutputField name="Final Score" feature="predictedValue" dataType="double" optype="continuous"/>
    </Output>
    <Characteristics >
     <Characteristic name="Merchant Category Code Score">
      <Attribute partialScore="-10">
        <SimplePredicate field="Merchant Category Code" operator"isMissing"/>
      </Attribute>
      <Attribute partialScore="20">
        <SimplePredicate field="Merchant Category Code" operator"equal" value="6011"/>
      </Attribute>
    </Characteristics>
     <Characteristics name="Shopping Transactions Score">
      <Attribute partialScore="0">
        <SimplePredicate field="Shopping Transactions" operator"isMissing"/>
      </Attribute>
      <Attribute partialScore="10">
        <SimplePredicate field="Shopping Transactions" operator"greaterOrEqual" value="1"/>
      </Attribute>
      <Attribute partialScore="20">
        <CompoundPredicate booleanOperator="and">
        <SimplePredicate field="Shopping Transactions" operator"greaterThan" value="1"/>
        <SimplePredicate field="Shopping Transactions" operator"lessOrEqual" value="4"/>
        </CompoundPredicate>
      <Attribute partialScore="30">
        <CompoundPredicate booleanOperator="and">
        <SimplePredicate field="Shopping Transactions" operator"greaterThan" value="4"/>
        <SimplePredicate field="Shopping Transactions" operator"lessOrEqual" value="12"/>
        </CompoundPredicate>
      </Attribute>
      <Attribute partialScore="50">
        <SimplePredicate field="Shopping Transactions" operator"greaterThan" value="12"/>
      </Attribute>
    </Characteristics>
     <Characteristics name="Amount Score">
      <Attribute partialScore="100">
        <SimplePredicate field="Amount" operator"isMissing"/>
      </Attribute>      <Attribute partialScore="15">
        <SimplePredicate field="Amount" operator"lessOrEqual" value="100"/>
      </Attribute>
      <Attribute partialScore="35">
        <CompoundPredicate booleanOperator="and">
        <SimplePredicate field="Amount" operator"greaterThan" value="100"/>
        <SimplePredicate field="Amount" operator"lessOrEqual" value="500"/>
        </CompoundPredicate>
      </Attribute>
      <Attribute partialScore="5">
        <SimplePredicate field="Amount" operator"greaterThan" value="500"/>
      </Attribute>
     <Characteristic>
    </Characteristics>
  </Scorecard>
</PMML>
```

A tabular representation of this scorecard would look like this:



Scorecard Variable	Value	Score
Merchant Category Code	nil (missing value)	-10
	6011	20
Shopping Transactions	nil (missing value)	0
	≥ 1	10
	> 1 ∧ ≤ 4	20
	> 4 ∧ ≤ 12	30
	> 12	50
Amount	nil (missing value)	100
	≤ 100	15
	> 100 ∧ ≤ 500	35
	> 500	5

IBM Safer Payments compatible scorecards

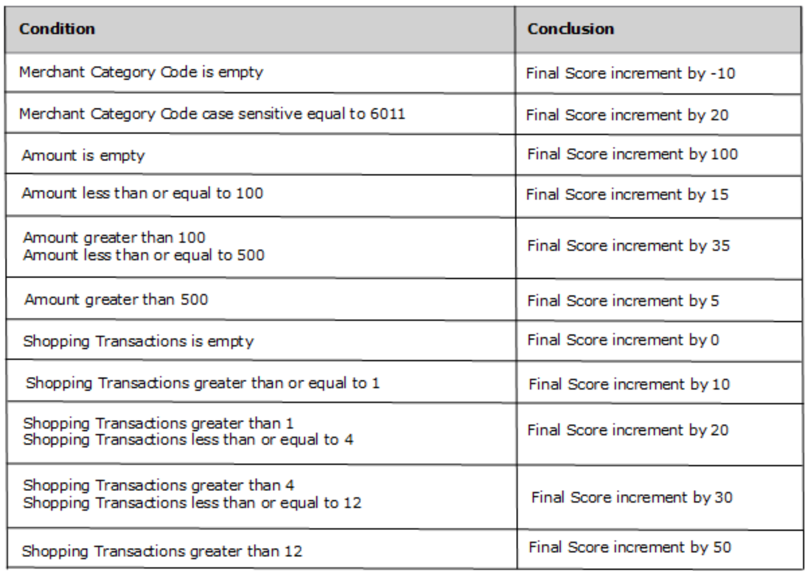
In order to be able to be imported into IBM Safer Payments a scorecard has to fulfill certain conditions:

- Attribute 'name' of every element 'DataField' has to correspond to an IBM Safer Payments model attribute name available to the ruleset where the scorecard shall be imported to.
- Attribute 'dataType' and attribute 'optype' of every element 'DataField' have to correspond to the respective IBM Safer Payments model attribute definitions. See PMML data types table in the model components online help for supported data types and mappings.
- Attribute 'name' of every element 'MiningField' has to correspond to an IBM Safer Payments model attribute name available to the ruleset where the scorecard shall be imported to.
- Attribute 'name' of every element 'OutputField' has to correspond to an IBM Safer Payments model attribute name that can be overwritten by conclusions of the ruleset where the scorecard shall be imported to.
- Attribute 'dataType' and attribute 'optype' of every element 'OutputField' have to correspond to the respective IBM Safer Payments model attribute definitions. See PMML data types table in the model components online help for supported data types and mappings.
- Valid values for the attribute 'operator' of every element 'SimplePredicate' are:
    - isMissing
    - isNotMissing
    - greaterThan
    - lessThan
    - greaterOrEqual
    - lessOrEqual
    - equal
    - notEqual

  The value has to correspond to the respective IBM Safer Payments model attribute definition (i.e. no 'greaterThan' operator for text values). Values not known to IBM Safer Payments will be imported per default as 'equal'.
- For the attribute 'booleanOperator' of element 'CompoundPredicate' only the value 'and' is being supported. When any other value is given, the whole rule will be ignored (not imported).

Remarks

- For supported PMML data types and their counterparts in IBM Safer Payments, see PMML data types table in the model components online help.
- IBM Safer Payments model attributes are only matched to scorecard variables if the attribute 'usageType' of element 'MiningField' is 'active'.
- If the scorecard is compatible to IBM Safer Payments, all **existing rules in the ruleset will be erased** and replaced by rules corresponding to the imported PMML file.
- In the example above the created rules would be:



Condition	Conclusion
Merchant Category Code is empty	Final Score increment by -10
Merchant Category Code case sensitive equal to 6011	Final Score increment by 20
Amount is empty	Final Score increment by 100
Amount less than or equal to 100	Final Score increment by 15
Amount greater than 100 Amount less than or equal to 500	Final Score increment by 35
Amount greater than 500	Final Score increment by 5
Shopping Transactions is empty	Final Score increment by 0
Shopping Transactions greater than or equal to 1	Final Score increment by 10
Shopping Transactions greater than 1 Shopping Transactions less than or equal to 4	Final Score increment by 20
Shopping Transactions greater than 4 Shopping Transactions less than or equal to 12	Final Score increment by 30
Shopping Transactions greater than 12	Final Score increment by 50

- Scorecards are usually only used for rules which change a score. Therefore all imported rules will have a conclusion with an 'increment by' operator, where the score value will be incremented by the value specified in the respective 'partialScore' attribute of an 'Attribute' element.

back to top

7.7.1.1.4 Rule Actions

Rule actions are performed on incoming transaction when all conditions of the rule are met. There are 5 different types of rule actions:

- **Add alarm**
  Adds an alarm for a case class. The data of an incoming transaction will be used to create an alarm for the selected case class. This is an alternative to define a conclusion with the meta attribute "case class".
- **Add entry**
  Adds an entry for a defined risk list. If an incoming transaction contains the input attribute of the selected defined risk list, an entry will be created with the transaction's attribute value. Further settings for the creation of defined risk list entries via rule actions can be found in the configuration of the respective defined risk list.
- **Add notification**
  Adds a notification. Choose a predefined notification that will be send when the rule fires. This is an alternative to define a conclusion with the meta attribute "notification".
- **Add reminder**
  Adds a reminder. Choose a predefined reminder that will be triggered when the rule fires. This is an alternative to define a conclusion with the meta attribute "reminder".
- **Set masterdata**
  Overwrites the stored value of the selected masterdata element. You can either use constant values (e.g. "5") or the value of an attribute of the current transaction by using the reference to this attribute (e.g. {CustomerName}).

Remark

It is possible to define multiple rule actions of the same type. If for example several actions of the type "Add alarm" for different case classes are defined, an alarm will be created for each case class. However using the same case class several times does not create multiple alarms for that case class and thus has the same effect as using it only once. Performing a "Set Masterdata" action on the same masterdata element several times, overwrites its value each time.

back to top

7.7.2 Decision Tree

Decision trees are one of the most standard and commonly used statistical models. The model is organized into hierarchical tree structure which combines multiple nodes. Scoring of a transaction against a decision tree starts from its root node and descends into child nodes. Each node has a logical predicate that determines the evaluation path i.e. processing is continued within a child which predicate is evaluated to true. This process is continued until a leaf node is reached. The latest score in the evaluation path serves as the predicted value for the evaluated message. Note that scores are defined by the attribute *score* on the tree nodes.

IBM Safer Payments supports both multi-node trees and binary trees. The type of the model is specified by the attribute *splitCharacteristic* of *TreeModel* element. The supported options for the *splitCharacteristic* attribute are:

- **multiSplit (default)**
  Each non-leaf node in the tree model may have an unrestricted number of children. If the attribute is not explicitly set, this value will be used as default value.
- **binarySplit**
  Each non-leaf node in the tree model should have exactly two children.

It might happen that none of child nodes evaluates to true while evaluating a tree model. In that case the optional *noTrueChildStrategy* attribute should indicate how to treat such situations (see section "No true child strategy" for the usage and supported options).

Score distribution

Scoring nodes in a decision tree define an attribute *score* which value serves as the predicted value if a transaction message chooses the node. In addition to the predicated value, PMML also allows to define so called *ScoreDistribution* element within each scoring node. This element specifies the probability distribution for each class that the model can predict. This means that scoring nodes can also compute probabilities of each class based on the record count voted for a particular class and total record count. For more information about score distribution refer to the official page of PMML standard.

Calculation of numeric predicates:

Note: Inside a PMML file a numeric constant in a predicate can have many decimals. Safer Payments will however allow a maximum of 8 decimals, so the decimal will potentially be cut off.

If a constant has more decimals than the attribute in the predicate, then a predicate can potentially return an unexpected boolean value for very large numeric attribute values.

Example: The predicate in PMML is defined as

<SimplePredicate field="AMOUNT" operator="lessOrEqual" value="0.0000068500000000002">

The AMOUNT attribute in this example has a length of 8 byte (64 bit) and 2 decimals. It can therefore store values up to $(2^{63} -1) / 10^{2} = 92,233,720,368,547,758.07$ (or the according negative values).

The decimal will be cut to *0.00000685*.

However since the predicate is compared with 64 bit integers and the decimal precision is 8 due to the constant, the predicate can only be guaranteed to return expected boolean results for absolute AMOUNT values less or equal to $(2^{63} -1) / 10^{8} = 92,233,720,368.54$

When the constant has less decimals, the AMOUNT values can be larger accordingly.

No true child strategy

Defined as *noTrueChildStrategy* attribute in *TreeModel* element, this attribute defines how to treat situations where none of the child nodes evaluates to true. The supported options for this attribute are:

- **Return null predication**
  Tree model returns no prediction.
- **Return last predication**
  In case one of the parent nodes scored already, the last score is taken, otherwise no prediction.

Ideally, every decision tree will provide a score. However, there might be situations, where the main predicate in a decision tree node evaluates to unknown. For situations like this, IBM Safer payments will always evaluate to false.

back to top

7.7.3 Neural Network

A neural network is a system of neurons arranged in neural layers. It comprises of one input layer, one or more hidden layers (usually simply referred to layers) and one output layer. All layers are interconnected and each neuron receives signals from neurons of the previous layer and processes them. IBM Safer Payments only supports execution of feedforward fully connected neural networks as a part of its model components.

Inputs and data transformations

The initial inputs are defined in the input layer. These are to be derived from transaction data by applying transformation to the attributes defined in the data field mapping section. Currently, we support the following transformations:

- **Field reference**
  Field references can be used to simply pass-through transaction data to the neural network without any transformation.
- **Normalize discrete values**
  Can be used to transform categorical string values into normalized discrete numeric values. If transaction value equals to the value defined in transformation, this returns 1.0 otherwise 0.0. The transformation value is defined as PMML attribute "value".

Activation of neurons

Each neuron activates the signal received from the previous layer. IBM Safer Payments supports all group 1 activation functions which take a linear combination X of weights, inputs and a bias. These activation functions are listed below:

- **threshold**
  activation = 1 if X > threshold otherwise 0
- **logistic**
  activation = $1 / (1 + e^{-X})$
- **tanh**
  activation = $(1 - e^{-2X}) / (1 + e^{-2X})$
- **identity**
  activation = X

- **exponential**
  activation = e^{X}
- **reciprocal**
  activation = 1 / X
- **square**
  activation = X^{2}
- **Gauss**
  activation = $e^{-(X^{2})}$
- **sine**
  activation = sin(X)
- **cosine**
  activation = cos(X)
- **Elliott**
  activation = X / (1 + |X|)
- **arctan**
  activation = 2 * arctan(X) / PI
- **rectifier**
  activation = max(0, X)

Outputs

The activations of the last (hidden) layer are the final scores of the neural network. These are mapped to IBM Safer Payments attributes as defined in the output field mapping section in the exactly same order in which neurons appear. The highest score always determines the predicted value. Notice that no normalization such as *simplemax* or *softmax* will be applied.

back to top

7.7.4 Random Forest

A random forest combines multiple decision tree models in one statistical model using segmentation approach. This approach implies that multiple tree models are grouped inside of a *Segmentation* element. Scoring takes place by evaluating each decision tree separately, independent of others and then aggregating all outputs across all decision trees. Hence, it is model generator's responsibility to specify how multiple models should be used and how scores should be aggregated by defining a multiple model method (see below for the usage and supported options).

Model combination methods

Defined as *multipleModelMethod* attribute in *Segmentation* element, this attribute defines an aggregation/combination method. The supported options are:

- **Majority vote**
  Selecting the score for which the highest number of tree models voted. For example, assuming that we have a random forest containing three trees and trees have predicted "NON_FRAUD", "FRAUD" and "FRAUD" respectively, the final output will be "FRAUD" (2:1). In case of two trees (two predictions "FRAUD" vs. "NON_FRAUD") this method can become ambiguous. The probablity of predicted values are calculated by the number of trees voted for this predicted value divided by the total number of trees.
- **Average and weighted average**
  Probabilities are computed as the (weighted) average of probabilities predicted by each model. The final predicted value would be the winning class which has the highest combined probability. The predicted probabilities are defined by a node element *ScoreDistribution* (see "Score distribution" section of decision trees for additional details).
- **Select first**
  Selecting the first model score. Having the above example with three trees, the final output in this case will be "NON_FRAUD".

back to top

8. Administration

This chapter covers the administration functions of IBM Safer Payments.

back to top

8.1 Mandators

IBM Safer Payments can be configured to operate sub-portfolios in a different way. For details, refer to the online help page on structural configuration. New mandators are created by clicking on the [New mandator] button.

If this capability is not needed, simply leave the default head mandator and do not define any more mandators here.

Notice that for any IBM Safer Payments function that is mandator specific, a choice of mandators is not shown (but implicitly selected) when there is only one mandator, regardless whether there is only one mandator defined or the user only has access privileges for one mandator.

back to top

8.1.1 Structural Configuration

Structural configuration involves empowering IBM Safer Payments to work for multiple sub-portfolios from within a single installation. This configuration is made by defining a mandator structure on this page.

Structured portfolios

In the past, most payment processing centre environments were designed to serve a homogenous portfolio of cardholders or merchants. All users could access all data, and one single model revision served all transactions. Even when the portfolio actually consisted of multiple sub-portfolios, the same users of the payment processing centre accessed all data, and if individualization of the model was required for different sub-portfolios, specific decision rules were introduced into the model. Previous generations of IBM Safer Payments served these needs well.

However, needs have changed. Today, most payment processing centres serve multiple portfolios, either of "internal" or "external customers". Customers in this sense can be lines of business of the same organization the processing centre belongs to ("internal"), or ("external") customers (issuers, acquirers) that are consumers for the processing services. What also has changed in the recent years is that fraud prevention has moved from a task that was perceived as a "black box" by the customers and typically performed by a few specialists of the payment processing centre, to a business process that customers want to control by themselves. Thus it has become commonplace that the customers of the payment processing centres require to access IBM Safer Payments functionality directly.

To cater for these new needs, earlier versions of the software introduced a "mandator" concept for structured portfolios, in which one installation could operate a portfolio consisting of sub-portfolios of multiple mandators, where each mandator could define its own model revision and work its own case classes.

IBM Safer Payments promotes this concept to an entire new level. IBM Safer Payments not only supports multiple mandators, it also supports any structure of mandators. Thus the payment processing centre can run a single IBM Safer Payments installation that serves a number of its mandators, yet each of its mandators can optionally also serve sub-mandators (and so forth). Each mandators can access IBM Safer Payments as if it is its own fraud prevention system, having its portfolio data "Chinese-walled" from the other mandators.

The mandator structure in IBM Safer Payments encompasses all aspects of fraud prevention. For instance the payment processing centre can define a "vanilla" fraud prevention model that deals with all standard fraud patterns. If a mandator feels that he needs to add (or subtract) any decision rules or other elements to (from) a model, he can do so without any effect on all other mandators. Since typically these "individualizations" involve only a small fraction of the settings of the "vanilla" fraud prevention model, much work is saved. The result is a standardized solution where any level of individualization can be made with minimum effort.

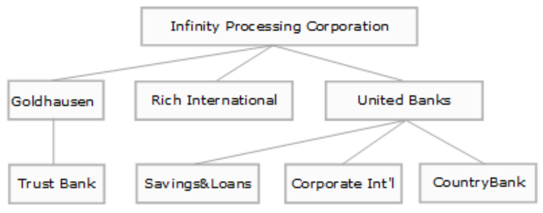
It should be noticed that using multiple IBM Safer Payments installations for a structured portfolio is a bad choice. Not only do administration and operation cost increase significantly, but the different installations cannot share important information in real-time. For instance, in a structured portfolio involving multiple issuers, if a merchant-side point of compromise is detected, this information would only be available with one IBM Safer Payments installation. And even if one would install cross-communication channels for such information, the problem remains that detecting certain fraud patterns is much harder (and much slower) when each IBM Safer Payments system only "sees" a fraction of the transaction volume. Therefore the only viable choice for a structured portfolio is to run fraud prevention on a single installation, so that the pattern recognition algorithm can use all data to verify fraud patterns.

Hierarchical mandator structure

IBM Safer Payments can be configured to support any number of mandators within a tree type structure. This structure follows a number of rules:

- A first mandator is already pre-defined when installing IBM Safer Payments. It is defined as the "top" node of the mandator hierarchy ("Infinity Processing Corp" in the example below).
- There may only be one top node in an IBM Safer Payments configuration. In situations where there are multiple "top-level mandators", the top node must be defined and considered a "virtual node", and the multiple mandators are defined right below it.
- For the definition of each subsequent mandator ("sub-mandator"), the next upper node of this new mandator must be selected as "head mandator" to form a hierarchy.
- Mandators may be added later at any position, even "in-flight", without IBM Safer Payments operation halting. Mandators may also be deleted, once all of its references in IBM Safer Payments have been deleted or deactivated (not top mandator).
- Because the mandator structure is key to all IBM Safer Payments operations, the hierarchy does not allow for moving of mandators to different heads.

For illustrational purposes, now assume the following example hierarchy with 8 mandators:



The example assumes that IBM Safer Payments is operated in the datacenter of Infinity Processing Corp that provides issuer and acquirer services for the 3 mandators: Goldhausen, Rich Int'l, and United Banks. Goldhausen itself owns Trust Bank. United Banks is a legal construction over Savings&Loans, Corporate Int'l, and CountryBank. United Banks thus has no own card portfolios.

There are a number of implications with such a hierarchy:

Inheritance

Within the mandator structure, a mandator below the top mandator inherits certain properties from mandators in the direct path "above" it. For instance, CountryBank inherits from United Banks and Infinity Processing Corp. Properties inherited include model revision elements, such as:

- attributes,
- profiling, and
- rules.

This allows defining model revisions in a "top-down" approach. For instance, all "standard" attributes would be defined with Infinity Processing Corp, so they are available with all mandators. Assuming that the three sub-mandators of United Banks require additional attributes, they would be defined with the model of United Banks. This way the additional attributes would be visible and available with United Banks, Savings&Loans, Corporate Int'l, and CountryBank (An "inherited" attribute can be used just like any "local" attribute. It, however, can only be edited from within the mandator it belongs to).

Typically "general purpose" profilings (e.g. "Revenue past 24h", "Number ATM withdrawals past 30 min at this ATM"), and thus their output attributes, would also be defined on a high hierarchy level – most likely the top mandator.

With rulesets and rules, inheritance can be used even more effectively by defining "general fraud rules" in the top mandator and rules addressing portfolio specific fraud patterns in mandators down the hierarchy; there can also be rules that revert a decision made by a rule in a higher node.

Mandator conditions

In a mandated configuration, it must be defined which mandator "owns" a transaction message/record. For this, each mandator is associated with so-called "mandator conditions". Typically these conditions involve IIN-Ranges ("Institute Identification Number") or other criteria that define "ownership" of records.

Mandator conditions are defined on the form that opens below this section when you select a mandator or create a new mandator using the respective button of the toolbar. They may use any attribute present in any of the mandator champions available to this mandator.

IBM Safer Payments interpretation of the mandator conditions follows the hierarchy: IBM Safer Payments starts with the top mandator and evaluates the conditions of all its sub-mandators. For each of these sub-mandators, IBM Safer Payments starts the evaluation of the sub-mandators of this mandator (if the conditions of sub-mandators of the same mandator are overlapping, IBM Safer Payments will compute all sub-mandators and continue their path).

Mandator ownership of a transaction message/record is essential for various access and computational decisions that IBM Safer Payments makes. These are described in detail below.

Computation

For any kind of IBM Safer Payments transaction message computation (online or batch), IBM Safer Payments must determine which model revisions are applied to it. IBM Safer Payments handles this the following way:

- Determine owning mandator (cf. above).
- Find path up to top mandator from owning mandator.
- Apply all champion models from top of the path (i.e. first top mandator, last owning mandator. This ensures that a sub-mandator is always computed after a mandator, ensuring that the more "further down" a mandator model is, the more it has the "last word" on decisions).

Because the owner mandator champion model is always computed last, it can overrule decisions made by head mandators.

While the principle that the lower mandator in the hierarchy always overrules the higher one is the correct one for most types of computation in a model, there is an exception. For instance, there may be rules for which the mandator up higher in the hierarchy should overrule the lower one. These rules are located in the "final rulesets" of a model revision. Since only certain applications do require this, whether or not final rulesets are available is defined on a per-mandator level in the mandator settings.

Data access

In structured portfolios data access is only granted to authorized users. Data access comprises – if the respective role privileges are granted – the following functions:

- view transaction details in query result tables and investigation pages,
- use data for simulation and analysis, and
- use data for rule generation.

In IBM Safer Payments, a user is authorized to view all data that is owned by the mandator and its sub-mandators the user is associated to.

For ease of use, in each of the functions listed above, the mandators whose data shall be included in the function can be selected in a checkbox list. For instance, if a user is associated with United Banks, he will be presented with the choice of United Banks, Savings&Loans, Corporate Int'l, and CountryBank. It is important to notice that the United Banks choice in this case comprises all transaction records that satisfy the United Banks conditions and none of the sub-mandators' conditions.

Mandator maintenance

A user that has *change* privileges for a mandator may create, change, and delete any sub-mandator "below". A user, however, may not change or delete a mandator if he has no *change* privileges of the head mandator.

This ensures that a user with *change* privileges for its own mandator cannot change the mandator conditions for example to expand its ownership of data.

There is one exception to this for the top mandator. A user with *change* privileges for the top mandator can change the top mandator's properties. Notice that the top mandator cannot be deleted and does not have data conditions.

Mandator deletion

Because mandator deletion can have disastrous effects when performed unintentionally, *change* privileges for mandators shall be granted only to users that understand these implications well. Even then, it is advisable to use a separate user account for such activities.

As a matter of precaution, the mandator deletion confirmation dialog requires the user to re-type the name of the mandator to ensure that the user is well aware of what he is doing.

Granted roles

The access and change privileges to entities associated to a mandator are controlled by "roles" that are granted to a user for each mandator separately. For details, refer to user accounts.

back to top

8.1.2 Mandator

Each mandator represents a logical (often also physical) unit that combines model revisions, access privileges, investigation, users, query and reports. It typically represents either a customer of a processor or a sub-portfolio. For details on IBM Safer Payments mandator hierarchies, refer to the online help page on structural configuration. Each of the counter output attributes is specified by a set of definitions that are made on this form:

Settings

- Head
  Determines (for a new mandator) to which existing mandator it is attached (as sub-mandator). Notice that you may not change this once a mandator is created.
- Name
  The name is used in all IBM Safer Payments pages and functions.
- Comments
  Comments are only for documentational purposes. It is advisable to fully comment the mandators.
- Remote lookup index
  This would typically be an index to the account (or PAN) attribute index. There are two reasons you may want to use a remote lookup index:
  - If you are using the IBM Safer Payments case investigation or query capabilities, you may flag (and unflag) individual transactions as fraudulent (or not). When this feature is used in a clustered configuration, the other instances must follow this flagging. For them to identify the correct transaction this index is used.
  - If you are using the IBM Safer Payments case investigation, you may enable reporting attributes in a case class when cases already exist for that case class. For the old cases, the reporting attribute value is determined based on their transaction. To speed up the process of determining the correct reporting attribute value on a remote instance, it is highly recommended to define this index.
- Final rulesets
  Enables an additional set of rules that are (1) computed *after* all other rulesets are computed, and (2) are computed in a sequence "upward" the mandator hierarchy. This implies that, unlike with *normal* rulesets, final rulesets of a higher mandator are computed after a lower mandator, and thus can override the decisions of mandators located lower in the hierarchy.
- Allow investigation user report
  This option enables evaluation of individual investigation user efficiency.
- Simulation memory
  Absolute maximum of total main memory that IBM Safer Payments allows for all simulations of all users for this mandator. Notice that IBM Safer Payments may decline simulation requests even when this limit is not reached, if the total memory on this IBM Safer Payments instance does not suffice to serve the simulation request. Please notice that the memory limit of a certain mandator may not exceed the memory limit of its head mandator.

Doublet detection

Lets you configure which transactions of this mandator shall be considered doublets:

- Enabled
  Activates doublet detection for this mandator.
- Include DDC
  Allows access to ddc values for the evaluation of doublets. This might have catastrophic effects on performance.
- Index
  Defines which index should be used for doublet detection. Usually, this should be the most distinctive index of your configuration.
- Attributes
  Choose a number of attributes, that characterize a unique transaction record. The number of attributes selected here should be

kept to a minimum as this effects computational performance.

Doublet detection analyses the sequence of an index entry. If it encounters a transaction records, that has identical values for all selected attributes, the currently processed transaction is discarded.

Status alarm indicators in dashboard

Lets you define how status alarms for users of this mandator are shown:

- Explanation text
  This text is shown below the header at all times.
- Explanation tooltip
  This text is shown as tooltip if the mouse pointer rests over the "Alarm" header.
- Online help type
  Lets you enable default or custom help texts in the header of the status alarm section.
    - custom online help only
      This option allows entering of a custom help text. Only the custom help text will be displayed.
    - custom and default online help
      This option allows entering of a custom help text. Both, the custom help text and the default help text, will be displayed.
    - no online help
      No help text will be displayed
    - default online help only
      Only the default help text will be displayed.
- Custom online help
  If you selected custom help text to be displayed (above), you may enter it here.

Mandator conditions

For all non-top mandators, these conditions define the data ownership.

Notice that the definition must comprise all data that shall be owned by sub-mandators of this mandator. Data that satisfies the mandator conditions of a sub-mandator is *not* considered to be owned by this mandator.

Mandator custom CSS layout configuration

IBM Safer Payments enables hierarchical customization of its user interface. All CSS definitions entered here are applied to this mandator and all its members (and their members).

back to top

8.1.2.1 Conditions

This element uses conditions. You can find further information in the conditions chapter:

10.4.1 Conditions

back to top

8.2 Roles

To complement the flexibility of IBM Safer Payments' mandator hierarchy, user privileges are managed using a role model that allows the definition and granting of roles for each mandator individually. In addition to this, each user account is tied to an ("associated") mandator, which also determines which data a user may access ("access" comprises the privilege to view individual transactions in query results or investigation screens, as well as in simulation, analysis and rule generation). The inner workings of this are explained in the remainder of this section.

Role definitions

The number of roles is not limited. Roles are defined for a specific mandator and can be inherited by sub mandators. Roles can be viewed/changed by users with the respective privileges. Refer to the help page of the section below for details on the definition of roles.

Grants

Once roles are defined, they can be "granted" to each user account assigned to a fitting mandator (depends on the setting whether or not roles are inherited). Refer to the online help page on "user accounts" for more details

back to top

8.2.1 Role

Each role is the combination of a number of privileges that are either enabled or not (check boxes). The privileges are organized in a hierarchy, where the privileges on lower levels of the hierarchy can only be enabled if the respective upper level privilege is enabled. To provide a clear overview of all enabled privileges, privileges that cannot be enabled because the upper level privilege is not enabled are hidden. All privileges displayed with "..." expand to additional privileges when checked. Right of the privileges, a more detailed explanation of the respective privilege is provided.

In most cases the upper level privilege is a "view" function privilege. Once this is enabled, either "change" or "add/edit/delete" privileges become selectable (and visible).

Whether the privileges superseding "view" are combined to "change" or split up to "add/edit/delete" privileges depends on the specifics of the respective IBM Safer Payments function.

Roles are associated with a mandator. Use the "inherit" option to make roles available with sub mandators as well. Otherwise, this role can only be used for setting mandator privileges of the mandator this role is associated with.

These privileges available in IBM Safer Payments are described on the page itself.

- **View dashboard**
  View the IBM Safer Payments dashboard with its status alarm indicators and KPI charts.
- **Reports...**
  View and generate reports that have already been defined.
  - **Change**
    Add, edit, and delete reports.
- **Investigation...**
  Investigation of cases generated for suspicious transactions.
  - **Query...**
    View and execute queries that have already been defined.
    - **Change**
      Add, edit, and delete queries.
      - **Change Extract Template**
        Add and edit extract template of a query.
  - **Common point query...**
    View and execute common point queries that have already been defined.
    - **Change**
      Add, edit, and delete common point queries.
  - **Create Cases**
    User may create cases right from a query result table.
  - **Fraud marking**
    Manual flagging of fraudulent transactions in transaction tables of investigation queries.
  - **Cases...**
    Investigate cases.
    - **Investigation supervisor...**
      User may see case selection and execute case search.
      - **Take over**
        Take over investigation cases that are reserved (follow up) for other users.
      - **Interrupt**
        Interrupt investigation cases that are currently worked by other users.
      - **View other users' cases**
        View cases that are being investigated by another user.
      - **Bulk transitions**
        Execute bulk case transitions via context menu.
    - **Change CPP**
      Add, edit, and delete CPP of cases.
    - **Send case actions**
      User may send case actions.
      - **Modify case actions**
        User may modify case actions before sending.
    - **Execute external queries**
      User may execute external queries.
  - **Masterdata**
    User may query masterdata.
  - **Change masterdata values**
    Change or enter masterdata in investigation cases or masterdata queries.
  - **Group by queries...**
    View and generate group by queries that have already beend defined.
    - **Change**
      Add, edit, and delete group by queries.
  - **CPPs...**
    View CPPs.
    - **Change**
      Add and edit CPPs.
- **Monitoring...**
  View Monitoring tab.

- **Compliance lists...**
  User may view the compliance list definitions.
    - **Change**
      User may change the compliance list definitions.
    - **Ad hoc check**
      User may perform compliance ad hoc checks.
- **Defined risk lists...**
  User may view the defined risk list definitions.
    - **Change**
      User may change the defined risk list definitions.
- **Defined risk list entries...**
  User may view the entries to defined risk lists.
    - **Import**
      User may import entries to defined risk lists.
    - **Change**
      User may change the entries to defined risk lists.
    - **Bulk delete**
      Bulk delete defined risk list entries.
    - **Bulk (de)activate**
      Bulk (de)activate defined risk list entries.
    - **View defined risk list audit trail**
      User may view defined risk list audit trail.

- **Model...**
  View model tab.
    - **Decision models...**
      View model revisions and their components.
        - **Edit...**
          Edit model revision.
            - **Take over**
              Take over a model revision that is reserved for other users.
            - **Data caches**
              Edit disk and memory data cache sizes.
            - **Indexes**
              Edit index definitions.
            - **Masterdata**
              Edit masterdata definitions.
            - **Modelling**
              Activation and configuration of modelling functions (test, simulation, analysis, and element generation).
            - **Inputs/outputs**
              Change model revision inputs and outputs.
            - **Change I/O encryption**
              Enable/disable encryption and purging of outdated entries of input/output attributes.
            - **Message mapping...**
              View mapping of message variables to model revision attributes.
                - **Change**
                  Add, edit, and delete message mappings.
            - **Mergings**
              Change mergings of messages to records.
        - **Copy**
          Copy model revisions.
        - **Initialize golive...**
          Initialize a new model revision to golive (mark challenger down to become champion).
            - **Confirm Golive**
              Confirm golive of a new model revision (golive of challenger to champion).
        - **Re-golive**
          Re-golive of a model revision that is already retired
        - **Retire champion**
          Retire a model revision in status champion without promoting another revision to status champion.
        - **Delete**
          Deletion of a model revision.
- **Administration...**
  Access administration tab.
    - **Mandators...**
      View mandator administration.
        - **Edit**
          Add, edit, and delete mandators.
    - **Roles...**

  View role definitions, their privileges, and mandator associations.
    - **Change**
      Add, edit, and delete role definitions.
- **Case actions...**
  View case action definitions.
    - **Change**
      Add, edit, and delete case action definitions.
        - **Change SQL**
          Add, edit, and delete case actions definitions that execute SQL commands.
    - **Run Test**
      Test case action definitions.
- **User groups...**
  View user group definitions.
    - **Change**
      Add, edit, and delete user groups.
- **Case states...**
  View case states.
    - **Change**
      Add, edit, and delete case states.
- **Case workflows...**
  View case workflows.
    - **Change**
      Add, edit, and delete case workflows.
- **Case close codes...**
  View case close code definitions.
    - **Change**
      Add, edit, and delete case close code definitions.
- **Case classes...**
  View case classes.
    - **Change**
      Add, edit, and delete case classes.
- **Working queues...**
  View working queues.
    - **Change**
      Add, edit, and delete working queues.
- **Case groups...**
  View case groups.
    - **Change**
      Edit and add case groups.
- **Notifications...**
  View alarm notification definitions.
    - **Change**
      Add, edit, and delete notification definitions.
        - **Change SQL**
          Add, edit, and delete SQL notification definitions that execute SQL commands.
    - **Run Test**
      Test notification definitions.
- **Text modules...**
  View text module definitions.
    - **Change**
      Add, edit, and delete text module definitions.
- **External queries...**
  View external query definitions.
    - **Change**
      Add, edit, and delete external query definitions.
    - **Run Test**
      Test external queries definitions.
- **Reminders...**
  View reminder definitions.
    - **Change**
      Add, edit, and delete reminder definitions.
- **Status alarm indicators...**
  View status alarm indicators.
    - **Change**
      Add, edit, and delete status alarm indicators.
- **Charts and KPI...**
  View chart and key performance indicator definitions.

        - **Change**
          Add, edit, and delete charts and key performance indicators.
    - **Charts and KPI...**
      View chart and key performance indicator definitions.
        - **Change**
          Add, edit, and delete charts and key performance indicators.
    - **Memory management...**
      View memory consumption of simulations.
        - **Stop simulations**
          Stop simulations of other users.
- **Cluster...**
  View Cluster Tab.
    - **Outgoing channel configuration**
      View outgoing channel configuration.
        - **Change**
          Add, edit, and delete outgoing channel configuration definitions.
            - **Change Basic Authentication**
              Edit settings of Basic Authentication.
        - **Run Test**
          Test outgoing channel configuration definitions.
- **Ticker...**
  View ticker.
    - **Add**
      Add ticker entries.
    - **Delete**
      Delete ticker entries.

back to top

8.3 User Accounts

The table below lists all user accounts for which you have access privileges.

Depending on your privileges, you may change (edit/add/delete) user accounts. You may also select from the toolbar whether or not disabled accounts shall be shown. Notice that depending on the IBM Safer Payments settings, you may or may not delete user accounts permanently (non-deletion is to ensure that all audit trails will lead back to a user account).

back to top

8.3.1 User Account

User administration

IBM Safer Payments is designed to support large scale applications with many hundred users belonging to a large number of mandators. For details on processing structured portfolios with mandator structures, visit the structural configuration help page.

IBM Safer Payments provides a two-dimensional privilege system allowing efficient but fine-grained control over which areas users can access and what operations they can perform:

- Global privileges
  These privileges are granted independently from the mandator structure. One of these global privileges is the "user account maintenance" privilege that enables a user to create/change user accounts. A user with this privilege can grant other accounts more privileges than he has for himself.
- Mandator privileges
  Each role is a set of privileges assembled for a specific "user role" in IBM Safer Payments. Roles are granted to user accounts always on a per-mandator basis. The privileges of the roles are thus only applied to this mandator.

Remarks

- Each user account is associated to one mandator. A user with the global privilege to change user accounts may only create user accounts for their associated mandator and its sub mandators.
- Each user account is associated to one mandator. A user's privilege to change their own mandator privileges applies to that mandator and its sub mandators.
- Roles can be passed on to sub mandators by enabling a checkbox next to the mandator-role association.
- A user that has been granted the role privilege to change or view one mandator also has the same privilege for its sub mandators even without explicitely passing the role to those sub mandators. This is an exception to all other mandator privileges that are strictly limited to the mandator granted.
- A user that has been granted the change privilege for a particular type of element (e.g. case actions) for one mandator is able to view this type of element for all mandators up the hierarchy. This is necessary to avoid creating duplicate elements on different levels of the hierarchy.

- A user's login may never be changed once the account has been created.
- For certain settings default values can be configured on the system configuration page to make setting up new accounts easier.

back to top

8.3.1.1 Global Privileges

These privileges are granted independently from the mandator structure as they cover access privileges to the general functionality of IBM Safer Payments:

- User accounts
  Defines which user account management actions may be performed by the user for other user's accounts.
- User self service
  Defines which user account self service actions may be initiated by the user (user changes settings of his own account).
- System configuration
  Defines which system configuration actions may be performed by the user. If the user has no rights for system configuration, the right 'export configuration' cannot be activated.
- Real-time intercept codes
  Defines which real-time intercept codes management actions may be initiated by the user.
- Messages
  Defines which message management actions may be initiated by the user.
- Cluster
  Privileges of this user account with respect to cluster management.
- Event log messages
  Privileges of this user account with respect to the configuration of event log messages.
- Jobs
  Privileges of this user account with respect to the configuration and execution of jobs.
- Password safes
  Defines which password safes management actions may be initiated by the user.
- Compliance list
  Privileges of this user account with regard to reloading compliance lists and searching/viewing compliance list entries.
- Key entry
  Defines if and which part of a public encryption key may be entered by the user.
- Key management
  Defines which key management actions may be initiated by the user.
- View system internals
  User may view the details of IBM Safer Payments' internal data structure.
- View unmasked data
  User may view encrypted data unmasked (if encryption is enabled).
- Change memory limits
  User may change memory limits of mandators.
- Export configuration
  User may export configuration.
- View system log messages
  User may view the system log messages.
- View audit log messages
  User may view the audit log messages.
- View transaction reports
  User may view transaction reports.
- View rules fired
  User may view the rules that created an alarm.
- View conditions of rules fired
  User may view the conditions of the rules that created an alarm.
- View manual icon
  View the manual icon in each help dialog.
- View private working queue
  User may view investigation cases of his private working queue.
- Set all user preference defaults
  If enabled, this user may set the defaults for all user's preferences.

back to top

8.3.1.2 Maintenance Functions

Maintenance functions are special API requests. Their functionality is assumed to not be used during standard IBM Safer Payments operations, but rather for highly specific maintenance functions, such as issues analyses, benchmarking, or facilitating automated testing. Maintenance functions can be enabled separately:

- Index functions
  Enables functions to reset an index (including all its related calendar profiles, events and masterdata) and to rebuild an entire index (erases existing one) using all records stored.
- Reset user preferences.
  Enables a function to reset preferences, searchfilter, table sizes and column orders for all users.
- Rewrite element to disk
  Enables a function to store a serializable object of an IBM Safer Payments installation on disk.
- Cleanout Revisions
  Enables a function that unloads all non-champion revisions from IBM Safer Payments and moves their file representation from the "cfg" to the "arc" directories of all IBM Safer Payments instances.
- Set MDC/DDC sizes
  Enables a function to change size of the specified xdc.
- Check health of index
  Enables a function to check specified index for issues. During execution of this maintenance function no other access to this index is possible.
- Reset FLI
  Enables functions to reset all outgoing FLI connections and rewind the FLI buffer's read position to the first unacknowledged FLI message.
- Cancel master key change
  Enables a function to cancel a master key change.
- Convert attribute data
  Enables a function to convert an integer value in format YYYYMMDDhhmmssZZZ to a timestamp in milliseconds since 1970-01-01 00:00:00.

back to top

8.3.1.3 Mandator Privileges

In this section, you may grant any number of roles to this user account for exactly one mandator. You may create multiple grants to associate multiple roles for one mandator or multiple mandators.

back to top

8.3.1.4 Simulation Memory

In this section, you may define how much memory can be used for simulation for a specific mandator. The user account may not use more simulation memory than defined with the mandator.

back to top

8.4 Case Actions

Case actions are messages sent by IBM Safer Payments to other systems during case investigation. Once case actions are defined, they can be configured for inclusion both for case classes and case close codes. (All these configurations are available from the administration tab.) Notice that you first have to define case actions, before they can be enabled for inclusion with case classes and case close codes.

With case close codes, the enabled case actions are automatically executed when a case is closed with the respective case close code.

With case classes, the enabled case actions are available for manual trigger (by the fraud investigator) during the investigation of a case.

The uses for case actions are manifold. Here are some examples:

- With a case closed as *fraud verified*, the respective case close code could also send a case action to the cardholder management system to close the account and invoke the embossing of a new card.
- During the investigation of a case, case actions can be used to send emails to the cardholder or merchant involved with the case.

back to top

8.4.1 Case Action

The definition of a case action involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  Allows you to temporarily enable/disable case actions without the need of redefining them or change model rules.
- **Name**
  Used to identify the case action. The name does not appear in the generated message.
- **Comment**
  Used to describe the case action. The comment does not appear in the generated message.

- **Mandator**
  Each case action belongs to the mandator of the selected outgoing channel configuration. Once created, mandator ownership does not change.
- **Outgoing channel configuration**
  The outgoing channel configuration that will be used to deliver the case action message. Outgoing channel configurations can be defined in the "cluster" tab and then referenced here. Depending on the type of the chosen outgoing channel configuration, the remainder of the form changes to display protocol specific settings. The values in the form will be pre-filled by default settings defined in the outgoing channel configuration, but offer to overwrite the values for the case action.
  - **SMTP**
    Case actions shall be sent by email using a SMTP type email service. SMTP case actions are queued and are sent out periodically (defined in IBM Safer Payments configuration) as batch. If the SMTP service is temporary unavailable, IBM Safer Payments attempts re-sending them also periodically. SMTP case actions are also stored on disk to ensure that unsent SMTP case actions will be attempted to be resent after a hard stop of IBM Safer Payments. This choice allows the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **Email "from" address**
      The sender address used for the outgoing SMTP case actions.
    - **Recipient address**
      Lets you choose between a "constant" recipient address (entered below) or taking the string value of the email meta attribute by choosing the option "variable from meta attribute" of the current transaction message. The latter allows for sending emails to individual cardholders, merchants, or acquirers.
    - **Constant email "to" address**
      If the recipient address is "constant", all outgoing SMTP messages are sent to this address. You may use multiple email addresses here, just separate them by semicolon.
    - **Subject template**
      Text template for the subject line of the outgoing SMTP message (see below).
    - **Body template**
      Text template for the message body of the outgoing SMTP message (see below).
    - **Support HTML formatting**
      When activated, mails send HTML formatted text as well as plain text (inside one message). Provide HTML formatted text through "HTML body template" box and plain text through "Body template" box.
    - **Encode HTML body base64**
      When activated, the HTML body text will be base64 encoded. Note that some more complicated HTML formattings might not be rendered correctly with base64 encoding. Please try with test notification first.
    - **HTML body template**
      HTML formatted text template for the message body of the outgoing SMTP message (see below).
  - **Message or HTTP Message**
    Case actions shall be sent by IP message to any other system. Analogous to SMTP case actions, message case actions are stored until they can be sent successfully. These choices allow the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **Content type (HTTP only)**
      The content type that will be used in the HTTP header. For additional information on how to use multipart forms, click here.
    - **Message template**
      Text template for the message (see below).
  - **File**
    Case action shall be stored as a file. There will be one file per each case action and the file name includes the system time with microseconds. This choice allows the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **File name prefix**
      Allows to define a file name prefix to distinguish messages from different case actions using the same outgoing channel configuration.
    - **Message template**
      Text template for the message (see below).
  - **SQL**
    Case action shall be executed as ODBC SQL. You have to install and configure a valid ODBC connector on all machines with active AMI. Make sure, that you can reach your database with your ODBC connector, before configuring ODBC SQL actions. The integration should be compatible with mySQL, postgreSQL and oracle ODBC connectors.

    It is not possible to parse return values or to import data by SQL into IBM Safer Payments.

    This choice allows the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **SQL Query**
      SQL template for the message (see below).
      This could be for example:

      ```
      INSERT INTO fraud VALUES ({PAN}, sysdate, 'REASON_1');
      ```

```
{call iris.Update_Status({PAN}, {Trx_Id}, 'REASON_2', {Trx_Time})}
```

- **Template file (.docx)**

  With word template file (.docx) case actions it is possible to generate template based word documents automatically from a case. First a suitable .docx template file has to be created with Microsoft Word (version 2010 or higher) and uploaded.

  Within the .docx template document it is possible to define placeholders for reporting attributes, query results, masterdata, user data case variables, and text modules which are filled when sending the case action. The values will be formatted according to their data type and "format" setting.

  You can apply different font styling options to your placeholders. Please note that you might need to choose a specific font family for some placeholders to make sure the filled text is rendered correctly. For example, if the replacement text contains Thai characters, font family for the corresponding placeholder could be set to "Browallia New" or to any other font family that includes Thai characters.

  When a template file (.docx) case action is sent from a case, the investigator may decide, if he would like to download the generated template file (.docx) document or if he prefers to attach the document directly to the case. To add a template file (.docx) document directly to the case, case attachments have to be activated for investigation.

Text templates

Within the text templates, you can define several placeholders for reporting attributes, query results, masterdata, user data case variables, and text modules (only for e-mail, sql, documents and Word documents) which are filled when sending the case action.

- **Reporting attributes**:
  Placeholders for reporting attributes are defined with curly brackets **{attribute name}**. You can use every attribute, that was defined when creating the case and which was reachable by the mandator. For cases created by an **index based evaluation** you can also use curly brackets to reference "Hit Condition Values" by their name.
- **Query results**:
  There are two different types of query result placeholders, depending on the type of case action they are used in.
  - **Placeholder for e-mail or sql-notifications:**
    Placeholders for query results are defined with curly brackets. They can only be used for case action previews and if query results were added to the case action previously. It is necessary to define the selected columns as attribute names in curly brackets too **{{attribute A}{attribute B}{attribute C}}**. This would create following result table:

attribute A	attribute B	attribute C
A1	B1	C1
A2	B2	C2

    The data has to be selected in case queries and added with right mouse click first. There will be no query result, if there was no selected data before sending the case action. Query results in e-mails will always be sent out as CSV file and will be visible in the e-mail body.
    For SQL Notifications, you can only use one attribute name between double curly brackets **{{attribute A}}**. An example for SQL would be

    ```
    UPDATE my_table SET column1='{{attribute A}}', column2='{{attribute B}}'
    WHERE column3='{{attribute C}}'
    ```

    If there are two entries added to the case action, this would perform two sql database updates.
  - **Placeholder for types File, HTTP and Message:**
    The template syntax, usage and resulting content are the same as for SQL and e-mail case actions. Instead of using HTML to format the table, its rows and columns are separated by characters which can be selected in the system configuration. An example could be to separate the columns using semicolons and the rows using a line break. The result would look like this:

    ```
    attribute A;attribute B;attribute C
    A1;B1;C1
    A2;B2;C2
    ```

  - **Placeholder for word template file (.docx) documents:**
    Placeholders for query results are defined by creating a table inside the Word document with at least one table row. Within the table header the selected columns can be defined with attribute names inside curly brackets. The first table row defines if the column will be filled with transaction data of the case action or not. In the case that the column should be filled, add an opening and closing curly bracket "{}" in the first table row in the respective column.
    This could be for example:

{TrxDateTime}	{Amount}	{Merchant ID}
{}	{}	{}

    Query results can only be used when query results were added to the case action previously. The data has to be selected in case queries and added via context menu. There will be no query result, if there was no selected data before sending the case action.
- **Case alarm tables**:
  The system configuration option "case aggregation history" causes cases to store a list of all aggregated alarms belonging to it. Those alarms behave identical to query results in that they can be added to the case action section and be referenced using the exact same placeholders. **However, it's not possible to add both case alarms and query results to case actions at the same time!**
- **Masterdata**:
  Placeholders for masterdata attributes are defined with double square brackets **[[masterdata attribute name]]**. You can use every masterdata, that is accessible by the mandator.
- **User data variables**:
  Placeholders for user data are defined with single square brackets **[InvestigatingUserName]**. You can switch between users by

changing the prefix

- [Investigating..]: The user, that is currently investigating the case.
- [Viewing..]: The user, that is viewing the case and sending the case action.
- [Closedby..]: The user, that closed the case.

The prefix has to be combined with a user variable name. For example, [..UserName] could be used as [InvestigatingUserName], [ViewingUserName] or [ClosedbyUserName]

- [..UserName]: The username as string.
- [..UserNameAndLogin]: The username, followed by the user login in parenthesis.
- [..UserUid]: The system internal user UID.
- [..UserEmail]: The users e-mail address.
- [..UserPhone]: The users phone number.
- [..UserLocation]: The users location.
- [..UserMandator]: The users mandator name.
- [..UserMandatorUid]: The UID of the users mandator.

- **Case variables**:
  Case variables are also defined with square brackets **[GeneratedOn]**. You can use following placeholders:
  - [CaseClass]: The name of the case class.
  - [CaseClassUid]: The UID of the case class.
  - [CaseClassId]: The ID of the case class.
  - [GeneratedOn]: The generation date as ISO formatted date.
  - [GeneratedOnTimestamp]: The generation date as UNIX timestamp.
  - [ClosedOn]: The case close date as ISO formatted date.
  - [ClosedOnTimestamp]: The case close date as UNIX timestamp.
  - [FollowupOn]: The followup date as ISO formatted date.
  - [FollowupOnTimestamp]: The followup date as UNIX timestamp.
  - [LastActionOn]: The last action date as ISO formatted date.
  - [LastActionOnTimestamp]: The last action date as UNIX timestamp.
  - [StateChangedOn]: The case state change date as ISO formatted date.
  - [StateChangedOnTimestamp]: The case state change date as UNIX timestamp.
  - [Score]: The case score.
  - [Hits]: The case hits.
  - [State]: The investigation state.
  - [StateUid]: The UID of the investigation state.
  - [ExtendedState]: The investigation state as visible in the case selection table.
  - [LastState]: The last investigation state.
  - [LastStateUid]: The UID of the last investigation state.
  - [FraudStatus]: The fraud status of the case close code, if the case was closed.
  - [CaseCloseCode]: The case close code, if defined.
  - [CaseCloseCodeUid]: The UID of the case close code.
  - [Mandator]: The case mandators name.
  - [MandatorUid]: The case mandators UID.
  - [CaseUid]: The case UID, as visible in the case selection table (1-123).
  - [CaseUidRaw]: The case UID, as visible in url or in file system (1000000000000123).
  - [Memo]: The text value of memo field.
  - [CaseAgeInDays]: The time since case generation in days.
  - [CaseAgeInHours]: The time since case generation in hours.
  - [CaseAgeInMinutes]: The time since case generation in minutes.
  - [DaysSinceLastAction]: The time since last action in days.
  - [HoursSinceLastAction]: The time since last action in hours.
  - [MinutesSinceLastAction]: The time since last action in minutes.
  - [DaysSinceStateChanged]: The time since case state changed in days.
  - [HoursSinceStateChanged]: The time since case state changed in hours.
  - [MinutesSinceStateChanged]: The time since case state changed in minutes.
  - [Firstparty]: Not compromised first party, only available for collusion type case classes.
  - [Counterparty]: Counterparty of the case, only available for collusion type case class.
- **Text modules**:
  Text modules are also defined with square brackets and the keyword **[TextModule]**. An investigator may choose one of the text modules when he sends a case action during case investigation. If he chose one, the placeholder of the message is replaced by the text template of the text module. Otherwise the placeholder is removed from the case action message. Notice, text module

placeholders can only be used in the body template. In the subject template, the placeholders for text modules are not replaced. Please notice as well that text module placholders can only be used for case actions with the target "SMTP" or "Document".

- **Loop construct for regular cases**:
  The loop construct allows to print out query results or alerted transactions, that were added to the case action section on the case investigation page in a flexible format in constrast to the query results placeholder that always returns a table. They are available for case action types "File", "HTTP", "Message", and "SMTP". The general syntax is:

  ```
  {[][]}
  ```

  The first set of square brackets contains a text template that can contain references to attributes, case variables, text modules and masterdata. This text template will be repeated and evaluated for each query result or alerted transaction. In case no data was added to the case action section, all alerted transactions will be iterated through. The second set of square brackets is optional and can contain a string that separates the outputs of each iteration. **Example:** Assuming we have a case for PAN "1111222211112222" with two alarms: one with an "Amount" value of "1.0" and another with a value of "2.0". We now could define a template like this:

  ```
  {[{PAN} | {Amount}][\n]}
  ```

  The result would be:

  ```
  1111222211112222 | 1.0
  1111222211112222 | 2.0
  ```

- **Loop construct for index based evaluation cases**:
  Index based evaluations that used multiple value evaluation can have several associated index nodes in one case. To retrieve information specific to those index nodes like the value itself or associated masterdata a loop construct was introduced for case actions of type "File", "HTTP", "Message", and "SMTP". The general syntax is:

  ```
  {[][]}
  ```

  The first set of square brackets contains a text template that can contain references to attributes, case variables, text modules and masterdata. This text template will be repeated and evaluated for each associated index node in the case. The second set of square brackets is optional and can contain a string that separates the outputs of each associated index node. **Example:** Assuming we have a case for customer "A" and accounts "1" and "2". We now could define a template like this:

  ```
  {[{PAN} | [[Account Type]]][\n]}
  ```

  Assuming account "1" is a private account and "2" is a business account, the result would be:

  ```
  1 | private
  2 | business
  ```

  Please note that the "Account Type" masterdata is defined on the "PAN" index and therefore changes in each iteration of the loop. **This is a special behaviour of index based evaluation cases.** In regular cases only the masterdata values associated with the current alarm (the one that is highlighted in the alerted transactions table) will be used.

- **Index based evaluation variables**:
  Index based evaluations offer special fields and placeholders:
  - [HitNodeValue]: The node value that triggered the alarm.
  - [HitAssociatedNodeValue]: The associated node value that triggered the alarm. If none is present the placeholder will be replaced with an empty string.
  - [CalendarComputationName]: Only usable in a loop construct (see above). Retrieves the calendar computation names for the current associated index node duplicating the line if needed.
  - [CalendarComputationValue]: Only usable in a loop construct (see above). Retrieves the calendar computation values for the current associated index node duplicating the line if needed.

Testing case actions

The [Save and create test case action] toolbar button above creates a sample case action. You can either use an existing case to fill the message template or create a case action using an empty template. Please note that reporting attributes, query results and masterdata will not be included in message templates when testing case actions.

back to top

8.5 Administration

This chapter covers the administration functions of IBM Safer Payments.

back to top

8.5.1 User Groups

The table below lists all user groups for which you have access privileges.

User groups can be used for configuring privileges of case transitions. The definition of a user group involves selection of multiple users from the list of potential case investigators. The selected users will be then privileged to execute certain case transitions on investigation cases. Once user groups are defined, they should be assigned to case transitions from the case class definition. The same user can belong to more than one group at the same time.

back to top

8.5.1.1 User Group

User groups have the following settings:

- **Name**
  Used to identify the user group.
- **Comment**
  Used to describe the user group. You may use this field to explain what this user group is used for.
- **Mandator**
  Each user group belongs to one mandator. Once created, mandator ownership does not change.
- **Users**
  Here you will find all users that are associated with the selected mandator or with submandators and have investigation privileges. The selected users will belong to the user group and will be privileged to execute certain case transitions on investigation cases if the user group is assigned to case transitions from the case class definition.

back to top

8.5.2 Case States

The table lists all case states that are defined and that you have access privileges for.

Case states are part of case workflow elements and allow to add new investigation stages. Case states are defined on a per mandator basis and are inherited downwards within the mandator hierarchy. An arbitrary number of case states can be created and once created, they can be used in case workflows (*Administration* -> *Case management* -> *Case workflows*). Cases are worked according to the case workflow definition and through the states selected from here.

Remarks

Notice that case states "New", "Closed" and "Followup" are default and thus cannot be changed or deleted.

back to top

8.5.2.1 Case State

The definition of case states involves a number of settings that are made in this form.

- **Name**
  Used to identify the case state.
- **Comment**
  Used to describe the case state. You may use this field to explain what this state is used for.
- **Mandator**
  Each case state belongs to one mandator. Once created, mandator ownership does not change.
- **Exclusive state**
  Determines whether or not the case state is exclusive. In exclusive state a case is assigned an investigator and can only be worked by that investigator. Cases that are in exclusive state are not available to other investigators. Any case transition to exclusive state will require an investigating user as manual input.

back to top

8.5.3 Case Workflows

The table lists all case workflows that are defined and that you have access privileges for.

IBM Safer Payments allows defining freely configurable case investigation workflows. Case workflows are created on a per mandator basis and are inherited downwards within the mandator hierarchy. Once created, case workflows should be used in case classes indicating how cases of each case class should be worked.

Definition

Case workflow definition consists of:

- case states, and
- case transitions that allow to switch between these states and apply certain actions on a case.

Each case transition has source states and a target state (from/to). Execution of a transition will change the state of a case to the target state specified in the transition settings. In addition to that, there are a number of other configuration parameters for case transitions.

Once case transitions are defined, they can be executed on cases:

- manually by the fraud investigator, or
- automatically if auto escalation conditions are defined and fulfilled.

Both case transition privileges and auto escalation conditions are configured from the case class settings when making case workflow selection.

Remarks

When creating a new case workflow, you need to make sure it contains the two mandatory states ("New" and "Closed") and valid transitions between the defined case states. Newly created cases are set to state "New". The final investigation state is "Closed", where cases are considered as closed. While these two states are mandatory in every case management workflow in IBM Safer Payments, it is possible to add an arbitrary number of investigation states in between.

Please notice that you need to update case class settings every time you add a new transition to case workflow because otherwise the newly created transitions will be available neither for manual nor for automatic execution.

If this capability of defining new case workflows is not needed, simply use the default case workflow which comes with IBM Safer Payments installation. Please note that the default workflow may not be changed.

back to top

8.5.3.1 Case Workflow

The definition of case workflows involves a number of settings that are made in this form.

- **Enabled**
  Allows you to temporarily enable/disable case workflows.
- **Name**
  Used to identify the case workflow.
- **Comment**
  Used to describe the case workflow. You may use this field to describe how case investigation will be carried out when using this case workflow.
- **Mandator**
  Each case workflow belongs to one mandator. Once created, mandator ownership does not change.
- **Case states**
  Lets you select case states that will be used in this case workflow.
- **Transitions**
  You may define one or more transitions between case states selected above.

back to top

8.5.3.2 Case Workflow Transitions

Case transitions allow to switch between case states. You can define case transitions below.
back to top

8.5.3.3 Case Workflow Transition

The definition of case transitions involves the following settings that are made in this form.

- **Name**
  Used to identify the transition.
- **Comment**
  Used to describe the transition.
- **Justification codes**
  Justification codes (aka "reason codes") are the choices fraud investigators have when executing case transitions. All codes entered here as free text will be available in the transition form for selection. You can leave this field empty if no justification is required for executing a certain transition.
- **Justification mandatory**
  If enabled, the investigator will be required to select a justification code (defined above) when executing the case transition.
- **Comment allowed**
  If enabled, the transition form shown to the investigator will contain an additional free text field where the user can enter his comments.
- **Comment mandatory**
  If comments for the transition allowed, determines whether or not the comment is mandatory.
- **Bulk transition**
  If enabled, the following transition can be executed by investigation supervisors (if appropriate privilege is granted) and working queue managers via context menu on a single case or on multiple cases in one step. Notice that if this option is not enabled, the transition will not be available for execution via context menu.
- **Source states**
  Lets you specify all possible case states from which this transition can happen.

- **Target case class**
  This option allows you to change the case class of a case when executing the transition. There are two options for this field available "current" and "definable".
  - **Current**
    Case class will remain the same when the transition is executed. If you do not want to change case class, simply leave selection on "current" and specify target state for the transition in the following field.
  - **Definable**
    This option will let you choose a target case class and a target state later from the case class definition when making case workflow selection.
- **Target state**
  If target case class "current" is selected, you need to specify target state for this transition.

back to top

8.5.4 Case Close Codes

The table lists all case close codes that are defined and that you have access privileges for.

Case close codes are the choices fraud investigators have when they close a case in investigation.

back to top

8.5.4.1 Case Close Code

Case close codes have the following settings:

- **Name**
  Name that will be shown as standard to the fraud investigator on the "case" page of case investigation.
- **Mandator**
  Each case close code belongs to one mandator. Once created, mandator ownership does not change.
- **Comment**
  Used to describe the case close code. The comment is also displayed to users that work investigation cases. You may thus use it to include case class specific investigation instructions.
- **Fraud status**
  Each case close code must map to one of the "principal" fraud statuses so that IBM Safer Payments can interpret case closings for its reporting and statistical analysis. Available fraud statuses are "fraudulent", "genuine", and "unknown".
- **Case actions**
  Lets you choose which case actions are triggered when a case is closed using this case close code.

back to top

8.5.5 Case Classes

The table lists all case classes that are defined for this mandator.

Case generation

In most applications of IBM Safer Payments, the reaction to a suspected fraudulent transaction is:

- intercepting with the transaction in real-time (typical means of interceptions are declining a transaction or referring it to a call center for security screening), and/or
- creation of an investigation case.

Both reactions are triggered by the model revision output attributes, and carried out by IBM Safer Payments according to its configuration.

With intercepting, the real-time interception codes are defined on the real-time interception codes page on the admin tab.

Alarms

The creation of investigation cases is a bit more complex. The first step is the creation of alarms. Each transaction message that is computed with a CaseClass meta attribute non-zero value is considered an alarm. Alarms are organized per mandator.

When an alarm is generated, IBM Safer Payments checks if there are unworked (new) cases for the "same payment entity" to which this alarm is consolidated. Because this check may take a while (a few milliseconds), this check is not performed within the (potentially real-time) message computation. Instead, each alarm generated is fabricated into a case (that looks as if there was no other case to consolidate with) and put into an "alarm" class for each mandator.

These not-yet-checked-for-consolidation cases are not displayed with any of IBM Safer Payments investigation functions. Instead, a scheduled "case loading" service periodically (as defined with IBM Safer Payments settings) works this "alarm" class for each mandator and checks for each alarm if it can be consolidated or not. If it can, consolidation is performed, if not, the case is put to the "case" class for the respective mandator so it becomes available to all IBM Safer Payments investigation functions.

Case consolidation

Case consolidation for regular cases (aka "aggregation") is performed whenever the new alarm hits on an entity (cardholder, merchant) for which an already unworked case exists. The case score and case class as well as the reporting attribute values of the "aggregated" case are taken from either the case or the alarm, whichever has the higher case score. If two alarms have the same case score, the newer one "wins" the consolidation, as it is assumed that "newer" means "more current" and thus "more accurate" to describe the case in the IBM Safer Payments investigation workflow. Collusion cases are always aggregated with respect to potential points of compromise. Compromised and affected first parties are consolidated and skimming time ranges are updated during case aggregation. If a case has already been worked on it is not used as a target for case consolidation.

Automated case transitions

IBM Safer Payments allows to automate execution of case transitions by defining a number of conditions on case variables and on reporting attributes as selected in case class setting. Conditions can be added to each case transition that is marked as automated. Scheduled jobs will constantly monitor cases of each case class and fitting cases will be escalated in a timely manner. You can also setup to send case actions with the execution of case transitions which could be used to notify investigation supervisors or to send reminders to investigators, for example. By default, all fitting cases are escalated every minute. You can change this setting from the system configuration page.

back to top

8.5.5.1 Case Class

The definition of a case class involves a number of settings that are made in this form. Two different types of case classes are available to provide specific information both for "regular" and "collusion" cases. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  Allows you to temporarily enable/disable case classes without needing to redefine them or change model rules.
- **Case class ID**
  If a transaction message is computed with a non-zero "case class" value, IBM Safer Payments checks if any of the case classes bears this number, and if so, creates an alarm for this case class. Please notice that you may not re-use numbers of inherited case classes.
- **Mandator**
  Each case class belongs to one mandator. Once created, mandator ownership does not change.
- **Name**
  Used to identify the case class.
- **Comment**
  Used to describe the case class. The comment is also displayed to users that work on a case of this case class. You may thus use it to include case class specific investigation instructions.
- **Case workflow**
  Each case class must define one investigation workflow. Cases of this case class will be modelled according to the selected workflow.
- **Type**
  There are two different types of case classes available. *Regular* case classes for "normal" investigation cases generated by rules and specific *collusion* cases generated by collusion processes.
  - **Regular**
    For regular case classes the following additional settings are available:
    - **Reporting attributes**
      Selects attributes to be displayed with the investigation case list. Typically you would choose attributes that described the case for users to quickly identify it. If you want to enable additional reporting attributes after cases have already been created for this case class, you should consider to use a remote lookup index (see mandator configuration for more information).
    - **Enabled for case creation**
      Allows you to use this case class for case creation. It will be possible to create cases with this case class even if the case class itself is disabled. In this case you'll have to enable the case class first in order to see the created cases for investigation.
    - **Column sequence**
      Lets you choose how reporting attributes shall be shown in the investigation case table. Notice that this table is shown when you select "case investigation" from the "investigation tab".
    - **Case consolidation**
      If enabled, alarms that have the same value as the aggregation attribute (selected below) for new cases are combined within the same case.
    - **Aggregation attribute**
      If case consolidation is enabled alarms that have the same value as the aggregation attribute are aggregated into once case. Only cases that have not been closed are used as targets for the alarm aggregation.
    - **Index based evaluation**
      For index based evaluation case classes the following additional settings are available:
      - **Case consolidation**
        If enabled, alarms that have the same value or values for the specified index (see below) are aggregated into an already existing and not closed case.
      - **Aggregation index**
        If case consolidation is enabled this field specifies which index is used to perform case aggregation. In a typical use case an index based evaluation would use a customer index as the primary index and an account index as the associated index. When enabling case aggregation using the associated index all associated index values are

evaluated meaning that an alarm for customer 1 with accounts A and B will only be aggregated to a case that also has accounts A and B and only those. **When choosing aggregation on the associated index, cases from index based evaluations that do not use an associated index will never be aggregated.**

    - **Highlight case alarms in queries**
      If enabled, all transactions belonging to a case will be highlighted in query results.

  - **Collusion**
    Collusion cases provide information about the detected (potential) point of compromise, compromised first parties and potentially affected first parties. Instead of reporting attributes there are additional case attributes, such as *First skimming transaction* and *Last skimming transaction*. During aggregation all such case attributes are updated. Collusion alarms are always aggregated for detected potential points of compromise.
  - **Aggregation trailing**
    Lets you enable or disable logging of alarm aggregation in case audit trail. If disabled, alarm aggregation logs will not be added to case audit trail.
  - **Limit the number of loaded audit trails**
    If enabled, allows you to specify the number of audit trail entries to be stored in memory. The remaining audit trail entries will be available on disk and can be loaded on demand from the case investigation page.
  - **Number of audit trail entries per case**
    The maximum number of audit trail entries per case to be stored in memory. If currently there are more audit trail entries in the memory than the number specified here, these entries will be dumped to the disk within the end of day job (being executed once a day). Please note that increasing this number will not result to immediate load of audit trail entries from disk but will only affect new audit trails to be added to a case. If you want the increased parameter to become effective for all existing cases, you need to restart the application.
  - **Queries**
    Lets you choose which of the index queries defined are automatically displayed on the case investigation "case" page for cases of this case class. Please note that only one fitting index query (with respect to counterparty and first party indexes) is displayed for potential points of compromise and affected first parties for cases of type *collusion*.
  - **Masterdata**
    Lets you choose which masterdata values are to be included with cases of this case class. For more information on masterdata, refer to the help page on the masterdata definition page. Please note that only masterdata attributes linked to the counterparty index are displayed for cases of type *collusion*.
  - **Case Actions**
    Lets you choose which case action values are to be included with cases of this case class.
  - **Case Close Codes**
    Lets you choose which close codes can be used to close a case of this case class.
  - **Case Transitions**
    This section contains all case transitions populated from the selected case workflow that can be executed on cases of this case class. Please note that transitions listed here need to be configured before you can use them in the case investigation module.

back to top

8.5.5.2 Case Transitions

Below you can find all case transitions of the selected case workflow. You need to update some of the settings by adding user groups and/or auto escalation conditions for each transition before using the transitions in the case investigation module.
back to top

8.5.5.3 Case Transition

Every case transition from the selected workflow needs the following settings to be defined.

- **Case actions**
  Allows you to select case actions that will be send automatically when this case transition is executed.
  - **Case close codes**
    Allows you to select case close codes, which can be used with this transition.
  - **User groups**
    Specify which user groups are privileged to execute this case transition. If left empty, only investigation supervisors and working queue managers will be privileged execute this case transition.
  - **Automated transition**
    If enabled, the following transition can be executed automatically by the system (if auto escalation conditions below are defined and fulfilled).
  - **Case close code**
    You need to specify a case close code for automated closing case transitions. Cases will be closed with the selected code.
  - **Followup user**
    You need to select an investigator for automated transitions to exclusive state. Cases will be added to the working queue of the selected user when the transition is executed.
  - **Auto escalation conditions**
    In addition to manual execution of case transitions, it is also possible to trigger case transitions automatically by defining a number of case conditions. Each condition will be evaluated, and cases that satisfy all conditions defined will be escalated. If you define no condition here, the transition will be executed.

Remarks

Please note that cases that have already been closed (case state "Closed") are not considered as escalation targets and thus no automated transition can be executed on these.

back to top

8.5.6 Working Queues

The table lists all working queues that are defined and that you have access privileges for.

Working queues provide a queuing and prioritization mechanism for investigation cases. Cases are constantly evaluated and sent off to working queues. By this way cases are becoming available to investigators assigned to the working queues.

Definition

Each working queue defines insertion conditions and selection of users assigned to the queue. Cases satisfying the insertion conditions are being added to the working queue. Investigators can then work cases by pulling them from the queues they are assigned to. Besides having assigned users, working queues might also have queue managers who besides doing case investgation, can also see the list of cases available in the working queue. In addition to that, working queue managers will be privileged to:

- Take over investigation cases that are reserved (follow up) for other investigators
- Interrupt investigation cases that are currently worked by other investigators
- View cases that are being investigated by another investigator
- Execute bulk case transitions via context menu

It is possible to prioritize working queues by assigning a numeric value to it. This prioritization indicator is used when evaluating cases, and when pulling a case from the queue.

Remarks

Notice that working queue prioritization only influences the sequence of working queues within a mandator. That means that first working queues of mandators on higher levels within the mandator hierarchy are evaluated.

back to top

8.5.6.1 Working Queue

Working queues have the following settings:

- **Enabled**
  Allows you to temporarily enable/disable the working queue.
- **Priority**
  Numeric value ranging from 1 to 10,000 indicating the priority of the working queue. Evaluation is performed in ascending order of the priorities. That means the higher the number, the LATER the working queue is evaluated within a mandator.
- **Name**
  Used to identify the working queue.
- **Comment**
  Used to describe the working queue. You may use this field to explain what this working queue is used for.
- **Mandator**
  Each working queue belongs to one mandator. Once created, mandator ownership does not change.
- **Cases from mandators**
  Restricts cases to be added to the working queue to the mandators specified.
- **Users**
  Selection of users assigned to the working queue. The assigned users will be able to pull investigation cases from the working queue.
- **Queue managers**
  Queue managers can see the list of all cases associated with the working queue. They will also be privilaged to perform all supervisor actions (taking over cases, interrupting, viewing cases of the working queue users and executing bulk transitions).
- **Show closed cases**
  Additionally, take on board closed cases (to inspect forepassed activities).
- **Insertion conditions**
  You may further restrict the cases to be added to the working queue to those whose reporting attributes and data satisfy certain criteria by defining conditions here.

back to top

8.5.7 Case Groups

Case Groups are used as a mechanism to assign CPPs to specific investigation cases, and that also means to assign to a mandator because a case group must be defined based on one mandator. Once a case group is defined it is available in the CPP selection application. When adding a new CPP the case group it belongs to must be selected. The table shows all case groups.

The following columns are always shown:

- **Mandator**
  Name of the mandator to which the case group belongs.
- **Name**
  Name of the case group.
- **Enabled**
  Indicates, if the case group is enabled.
- **Comment**
  Comment of the CPP.
- **Inherit to submandators**
  Indicates, if the case group inherits to submandators. If this is the case, submandators of the listed mandator are able to view and create CPPs of/for the case group.
- **Evaluation attribute**
  An attribute of closed fraudulent cases that are connected with a CPP. It will be added up and displayed in the CPP table.

Remarks

- Case groups must be defined and enabled to create and use CPPs.
- Notice that you can sort the investigation table by clicking on column headers. To sort for more than one column, simply click the columns in sequence (the former "inner" sorting will remain). Sorting preferences are stored with your user's account.

back to top

8.5.7.1 Case Groups

The configuration of case groups.

- **Enabled**
  A case group can be enabled or disabled. CPPs cannot be assigned to disabled case groups. If a case group that contains CPPs gets disabled, the belonging CPPs are not displayed in the CPP table anymore and cannot be assigned to cases.
- **Inherit to submandators**
  CPPs of case groups that inherit to submandators are visible to submandators. Furthermore, submandators are able to assign CPPs to the case group and view and change belonging CPPs.
- **Name**
  The name of the case group.
- **Comment**
  A comment which is added to the case group. Comments do not influence computation and are informational only.
- **Mandator**
  The mandator the case group belongs to.
- **Evaluation attribute**
  Choose an attribute that will be added up and displayed in the CPP table, if it belongs to a case that was closed as fraudulent and is connected with a CPP. The CPP list displays the sum of values in the column 'Sum of evaluation attribute' from all the cases to which the CPP is assigned to.

back to top

8.6 Notifications

Notifications (typically sent via email, mobile text, or fax media) are a potential reaction to a transaction message. IBM Safer Payments creates a notification the same way as it generates alarms for investigation cases.

There are many similarities to case classes. Notifications are defined on a mandator basis and are triggered by the "notification" meta attribute value being non-zero. The value of the "notification" meta attribute determines which of the notifications defined is generated for the transaction message.

Notice that notifications can be generated in parallel to cases, reminders and real-time reactions, such as to intercept a transaction.

back to top

8.6.1 Notification

The definition of a notification involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  Allows you to temporarily enable/disable notifications without the need of redefining them or change model rules.
- **Notification ID**
  Value of the "notification" meta attribute that triggers this notification.
- **Name**

Used to identify the notification. The name does not appear in the generated notification message.

- **Comment**
  Used to describe the notification. The comment does not appear in the generated notification message.
- **Mandator**
  Each notification belongs to the mandator of the selected outgoing channel configuration. Once created, mandator ownership does not change.
- **Outgoing channel configuration**
  The outgoing channel configuration that will be used to deliver the notification message. Outgoing channel configurations can be defined in the "cluster" tab and then referenced here. Depending on the type of the chosen outgoing channel configuration, the remainder of the form changes to display protocol specific settings. The values in the form will be pre-filled by default settings defined in the outgoing channel configuration, but offer to overwrite the values for the notification.
    - **SMTP**
      Notification shall be sent by email using a SMTP type email service. SMTP notifications are queued and are sent out periodically (defined in IBM Safer Payments configuration) as batch. If the SMTP service is temporary unavailable, IBM Safer Payments attempts re-sending them also periodically. SMTP notifications are also stored on disk to ensure that unsent SMTP notifications will be attempted to be resent after a hard stop of IBM Safer Payments. This choice allows the following entries:
        - **Format values**
          Format/unformat values which are inserted in the message template.
        - **Email "from" address**
          The sender address used for the outgoing SMTP notification.
        - **Email "to" address type**
          Lets you choose between a "constant" recipient address (entered below) or taking the string value of the "email" meta attribute of the current transaction message. The latter allows for sending emails to individual cardholders, merchants, or acquirers.
        - **Constant email "to" address**
          If the recipient address is "constant", all outgoing SMTP messages are sent to this address. You may use multiple email addresses here, just separate them by semicolon.
        - **Subject template**
          Text template for the subject line of the outgoing SMTP message (see below).
        - **Support HTML formatting**
          When activated, mails send HTML formatted text as well as plain text (inside one message). Provide HTML formatted text through "HTML body template" box and plain text through "Body template" box.
        - **Encode HTML body base64**
          When activated, the HTML body text will be base64 encoded. Note that some more complicated HTML formattings might not be rendered correctly with base64 encoding. Please try with test notification first.
        - **Body template**
          Text template for the message body of the outgoing SMTP message (see below).
    - **HTML body template**
      HTML formatted text template for the message body of the outgoing SMTP message (see below).
- **Message or HTTP Message**
  Notification shall be sent by IP message to any other system. Analogous to SMTP notifications, message notifications are stored until they can be sent successfully. These choices allow the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **Content type (HTTP only)**
      The content type that will be used in the HTTP header. For additional information on how to use multipart forms, click here.
    - **Message template**
      Text template for the message (see below).
- **File**
  Notification shall be stored as a file. There will be one file per each notification and the file name includes the system time with microseconds. This choice allows the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **File name prefix**
      Allows to define a file name prefix to distinguish messages from different case actions using the same outgoing channel configuration.
    - **Message template**
      Text template for the message (see below).
- **SQL**
  Notification shall be executed as ODBC SQL. You have to install and configure a valid ODBC connector on all machines with active AMI. Make sure, that you can reach your database with your ODBC connector, before configuring ODBC SQL actions. The integration should be compatible with mySQL, postgreSQL and oracle ODBC connectors.

  It is not possible to parse return values or to import data by SQL into IBM Safer Payments.

  This choice allows the following entries:
    - **Format values**
      Format/unformat values which are inserted in the message template.
    - **SQL Query**
      SQL template for the message (see below).

This could be for example:

```
INSERT INTO fraud VALUES ({PAN}, sysdate, 'REASON_1');
```

```
{call iris.Update_Status({PAN}, {Trx_Id}, 'REASON_2', {Trx_Time})}
```

Text templates

Within the text templates, each transaction record attribute value can be filled in. Just put the attribute name (as it appears in the model) in curly brackets. IBM Safer Payments automatically fills in the appropriate value and formats it according to the attribute settings. When using a notification inside an **index based evaluation** all special template placeholders for index based evaluations are available.

Testing notifications

The [Save and create test notification] toolbar button above creates a sample notification. You can either use a specific record to fill the defined message template, or create a notification using an empty template.

back to top

8.7 Text Modules

Text modules can be used to predefine text templates which can be added to case actions. IBM Safer Payments provides case actions to send messages to other systems during case investigation. For the case actions types "SMTP", "SQL" and "Word (.docx)" a placeholder for text modules can be defined by adding the keyword [TextModule] into the body template. An investigator may choose one of the text modules when he sends a case action during case investigation. If he chose one, the placeholder of the message is replaced by the text template of the text module. Otherwise the placeholder is removed from the case action message.

The table below lists all defined text modules. To add a new text module click [new text module]. To view or change text modules, left click on the respective row in the table.

back to top

8.7.1 Text Module

The definition of an text module involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  Allows you to temporarily enable/disable text modules.
- **Name**
  Used to identify the text module. The name is used to choose the text which should be added to the case action.
- **Comment**
  Used to describe the text module.
- **Mandator**
  Each text module belongs to one mandator. Once created, mandator ownership does not change.
- **Text template**
  Used to define a text which can be added to a case action (of the type SMTP or document) during case investigation.

back to top

8.8 External Queries

External queries are means of requesting data from another system for case investigation purposes. They are used to ensure that the data used to evaluate a case is up to date.

External queries can be sent as IP or HTTP messages and are triggered either while loading a case or upon user request.

External queries within IBM Safer Payments are modelled using the same template mechanism as case actions or notifications. The response is parsed according to the definition and displayed within the case. It is even possible to update the stored data of the case using the received data.

back to top

8.8.1 External Query

The definition of an external query involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. Settings are:

- **Enabled**
  Allows you to temporarily enable/disable external queries without the need of redefining them.
- **Name**

Used to identify the external query. The name does not appear in the generated message.

- **Comment**
  Used to describe the external query. The comment does not appear in the generated message.
- **Outgoing channel configuration**
  The outgoing channel configuration that will be used to deliver the external query. Outgoing channel configurations can be defined in the "cluster" tab and then referenced here. External queries support outgoing channel configurations of the types "HTTP" and "message".
- **Mandator**
  Each external query belongs to one mandator defined in the outgoing channel configuration. This mandator can not be changed.
- **Format values**
  Format/unformat values which are inserted in the message template.
- **Content type (HTTP only)**
  The content type that will be used in the HTTP header.
- **Message template**
  This is the template which is used to generate the request. You can use any reporting attribute within IBM Safer Payments by just putting its name within curly brackets. IBM Safer Payments will replace that part of the message with its actual value from the case.
- **Response mapping**
  IBM Safer Payments expects XML responses. This setting tells IBM Safer Payments which XML-tag will be used to fill which reporting attribute. It is defined as a list of pairs, where each pair is separated by a ';'. The values are expected as 'attributename:XML-tag', where attributename is the name of an attribute as defined in IBM Safer Payments and XML-tag is the name of the tag without '<' or '>' characters.
- **Overwrite case values**
  The mapped response values can be used to overwrite the values stored within the case.

back to top

8.9 Reminders

Reminders are timers that are triggered similarly to cases and notifications. Reminders are defined on a mandator basis and are triggered by the "reminder" meta attribute value being non-zero. The value of the "reminder" meta attribute determines which of the defined reminders is triggered for the transaction message.

A reminder works as follows:

- Once the reminder is triggered, it waits the defined time period.
- Once this time period is up, it re-computes the triggering transaction messages as if it was sent at that moment.
- Attribute values of all outputs are reset to "nil".
- The value of the "Message type ID" meta attribute is overwritten with the value defined in the reminder. This is used to differentiate the "first" computation of the transaction message (when it originally came in) from the "second" computation of the transaction record (when the reminder expired).

Remarks

- Make sure that the rules that trigger a reminder will not fire for the MTID value defined in the reminder. Otherwise the reminder would be re-triggered and executed repetitively.
- Reminders are only executed on the primary IBM Safer Payments instance that received the initial transaction message.
- There is no new transaction record produced by the execution of the second computation. Also the transaction message remains the same with the transaction record stored.
- Reminders are not stored permanently. When you shut down the IBM Safer Payments instance or when the IBM Safer Payments instance stops operation, the reminders not yet executed will be deleted.

Testing reminders

The [Save and create test reminder for most recent record (last URID)] toolbar button creates a sample reminder for the most recent record generated.

back to top

8.9.1 Reminder

The definition of a reminder involves a number of settings that are made in this form. Rest the mouse pointer over a setting for details. The following settings are available:

- **Enabled**
  Allows you to temporarily enable/disable reminders without the need to redefine them or change model rules.
- **Reminder ID**
  Value of the "reminder" meta attribute that triggers this reminder.
- **Name**
  Used to identify the reminder. The name does not appear in the generated reminder message.

- **Comment**
  Used to describe the reminder. The comment does not appear in the generated reminder message.
- **Mandator**
  Each reminder belongs to one mandator. Once created, mandator ownership does not change.
- **Message**
  The triggering transaction message is computed again with this message after the reminder has expired.
- **Time**
  Waiting time period (in seconds) of the reminder before it recomputes the transaction message.

back to top

8.10 Master Keys

The table lists all master keys for which a public key is available within IBM Safer Payments.

Notice that you may not add master encryption keys from this user interface for security reasons.

To add (master) keys, add your new generated usage key files ("key_<key_id_n>.iris") to the IBM Safer Payments "key" folder on all instances and press the [Reload private keys from disk] button.

back to top

8.10.1 Encryption Keys

The table lists all encryption keys for which the public key is available within IBM Safer Payments. To enter public keys or to activate keys, click on the key row in the table.

Notice that you may not add private encryption keys from this user interface for security reasons. Private keys must be installed by the administrator at file level. Pay attention to the timed validity of keys. For more details, read the background help page for PCI DSS encryption of IBM Safer Payments.

back to top

8.10.2 Encryption Key

Which functions are available depends on your specific privileges:

- **Key entry**
  Depending on the privileges of the roles that your user account is granted, you may either enter the left key or right key. You may also add a comment to your entry that remains visible with the key table. Notice that each entry overwrites any previous entry. There is therefore no editing of existing entries. Notice that the key pair ID is defined with the private key and therefore cannot be changed.
- **Key validation**
  Use the toolbar icon to check the validity of a key triplet once both public keys are entered.
- **Key activation**
  Also checks validity of key and activates key triplet if key was valid. The previously valid key is automatically disabled.
- **Key revocation**
  Removes all key triplets from IBM Safer Payments permanently. Keys cannot be activated again after revocation. Notice that you can also revoke keys that are not yet activated or for which the public keys have not (yet) been entered.
- **Change master key**
  Re-encrypts all data with a new master key. The re-encryption process takes a long time and all instances will be inactive during this process.

For more details, refer to the background help page for PCI DSS encryption of IBM Safer Payments.

back to top

8.11 Job Schedule

The table lists all jobs that have been defined. Because of the status information provided an auto-refresh time interval (on the "system configuration" page) can be defined which causes this page to reload periodically.

To view or change job details and parameters, left click on the respective row in the table.

Use the context menu to execute or stop one (or in new user interface multiple) job(s), or to generate reports when report generation jobs are selected.

For more details, read the background help page on IBM Safer Payments interfaces that is available from the main help page.

back to top

8.11.1 Job

Each job represents a manually or periodically executable task. There are different types of jobs available. The following settings are common for all of them:

- **Name**
  Used to identify the job.
- **Comment**
  Used to describe the job.
- **Job type**
  Select which kind of job you would like to define (see below for type specific settings). Currently these types are available:
  - **Load file**
    The standard way of loading a batch of files into IBM Safer Payments. For more details, read the BDI (batch data interface) overview.
  - **Generate report**
    Job type to generate reports. The results can be downloaded using the button at the top of the form.
  - **Evaluate sanction list**
    Job type used to perform offline compliance checks.
  - **Execute index based evaluation**
    Job type used to execute index based evaluations. **Important:** While index based evaluations are executed, the used indexes are locked against changes, so message computations affecting those indexes will be halted until the job finishes. Because of this, index based evaluation jobs should be run on an instance that does not process incoming transactions.
  - **Export data**
    Job type used to export transactional data.
- **Priority**
  The priority for the operation system to execute the job. Notice that the maximum priority is limited by the general IBM Safer Payments setting (system configuration page). The higher this priority, the more computational resources will be given by IBM Safer Payments to the computation of this job.
- **Maximum threads**
  Maximum number of parallel threads (if available from the overall BDI thread pool size as configured) that IBM Safer Payments attempts to employ for the processing of this job. The higher this number, the more parallel computing resoures will be given by IBM Safer Payments to the computation of this job. If the total number of currently available BDI threads (as configured on the system configuration page minus the ones currently used by other jobs) is lower than the maximum number of threads set here, IBM Safer Payments will only use the available amount of threads for the entire duration of its job processing.
- **Recurrence**
  Select if a job should be executed manually, daily, weekly, monthly or periodically using a defined time interval. Notice that for jobs repeated in a defined time interval, IBM Safer Payments sets the next time the job runs in alignment to full hours. If you, for instance, create a job at 15:43, the next run would be at 16:00, if you set the time interval to 20, 30 or 60 minutes. If you set the interval to 10 minutes, the next run would be at 15:50.
- **Suspended**
  If checked, the recurring automatic execution of this job is suspended; it can still be executed manually.
- **Day of week**
  Select on which day of the week (server time) a weekly job shall be executed.
- **Day of month**
  Select on which day of the month (server time) a monthly job shall be executed. Numbers 1 to 31 are valid. If the chosen day doesn't exist in a given month, for example 30 and 31 in February, the job is executed on the last day of the month.
- **Daytime**
  Select the server time at which daily, weekly or monthly jobs shall be executed.
- **Repeat every**
  Specifies the interval after which a job should be restarted.

The following settings are only available for job type "Load file":

- **Wait for semaphore file**
  If selected, execution of a job waits until a semaphore file is delivered to ensure that partial file deliveries are not processed.
- **Create log file**
  If checked, a log file is created during file processing; the log will contain a row for each record containing the (XML formatted) response of the computation.
- **Curtail masterdata**
  If checked, all masterdata elements, that have an MTID meta attribute insert condition for the message type ID of this job and are not updated with this job, are deleted (set to nil value) upon completion of the job.
- **Parameter**
  Customer specific processing options can be entered here. These options are provided by the IBM Safer Payments team.
- **Message**
  This message will be associated with all messages coming from this job. The meta attribute "Message type ID" will automatically be filled with the respective MTID of the selected message.
- **Continue after error**
  If enabled, rows in which format errors are detected will be skipped and event log messages will be generated; if disabled, job halts at error
- **Check row length**

Only available for messages using the FCD format. If enabled, only rows that have the expected length will be processed. The expected length is computed using the defined mappings.

- **Incoming directory**
  Directory for the files to be loaded by this job. According to the type of the selected message ("CSV format", "FCD format" or "XML format" or "Nested XML format") all files with suffix .csv, .fcd, .xml or .txt within the incoming directory are imported in alphabetical order during the job. In case you want to send other formats, please choose the custom message type and make sure you have the custom parser library in place.
- **Archive directory**
  Directory to which files that have successfully been loaded into IBM Safer Payments are moved
- **Error directory**
  Directory to which files that have not successfully been loaded into IBM Safer Payments are moved
- **Enable encrypted delivery**
  If selected, delivery of encrypted job files is activated. Job files need to be encrypted with the key from encrypted aes key path. You will find further details for the encrypted job import in Importing encrypted job files.
- **Password safe**
  Choose the activated rsa decryption key from password safe here. This key will be used to decrypt the encrypted aes key.
- **Aes key path**
  Path to the encrypted aes key, which is used to decrypt the job file(s). The key itself will be decrypted with the activated rsa key from password safe.
- **Re-create interval index**
  If enabled, selected interval indexes will be reset and recreated during the execution of the job. Use this feature to update interval indexes while loading masterdata. The recreation will be performed on each instance.
  - **Mandators**
    Select the mandators whose indexes should be recreated.
  - **Indexes**
    Select the interval indexes that should be recreated.
  - **Remote wait factor**
    The relative wait factor for a remote recreation to get transmitted. The first recreation of a remote instance is triggered immediately after the primary recreation job. The second remote recreation will wait for the time of the primary recreation job, multiplied by this wait factor.
    "0" for immediate transmission to all instances after successful execution of the recreation job. "1" for a cascaded execution of the recreation. It is recommended to use "1.05" to have an additional wait of 5% to avoid having simultaneous recreations on 2 machines.

The following settings are only available for job type "Generate report":

- **Mandators**
  Select the mandators that should be included in the report generation job.
- **Reports**
  Select the reports to be executed for the report generation job. Currently, group by queries, merchant monitoring rules and reports from the report section that are defined for the selected mandators will be available for selection here.
- **Use outgoing channel configuration**
  If checked, the report(s) will be sent via outgoing channel configuration when the job is executed.
  - **Outgoing channel configuration**
    The outgoing channel configuration that will be used to deliver the report(s). Outgoing channel configurations can be defined in the "cluster" tab and then referenced here. Depending on the type of the chosen outgoing channel configuration, the remainder of the form changes to display protocol specific settings. The values in the form will be pre-filled by default settings defined in the outgoing channel configuration, but offer to overwrite the values for the job.
    - **SMTP**
      SMTP messages are queued and are sent out periodically (defined in IBM Safer Payments configuration) as batch. If the SMTP service is temporary unavailable, IBM Safer Payments attempts re-sending them also periodically. SMTP messages are also stored on disk to ensure that unsent SMTP messages will be attempted to be resent after a hard stop of IBM Safer Payments. This choice allows the following entries:
      - **Email "from" address**
        The sender address used for the outgoing SMTP notification.
      - **Email "to" address type**
        Lets you choose between a "constant" recipient address (entered below) or taking the string value of the "email" meta attribute of the current transaction message. The latter allows for sending emails to individual cardholders, merchants, or acquirers.
      - **Constant email "to" address**
        The recipient address is "constant", all outgoing SMTP messages are sent to this address. You may use multiple email addresses here, just separate them by semicolon.
      - **Subject template**
        Text template for the subject line of the outgoing SMTP message.
      - **Body template**
        Text template for the message body of the outgoing SMTP message.
    - **File**
      There will be one file per each report and the file name includes the system time with microseconds. This choice allows the following entries:
      - **File name prefix**
        Allows to define a file name prefix to distinguish messages from different jobs using the same outgoing channel

configuration.

The following settings are only available for job type "Evaluate scanction list":

- **Mandators**
  Select the mandators from which compliance lists will be selectable.
- **Compliance lists**
  Select the compliance lists for the sanction list job. Only offline compliance lists defined for the selected mandators will be available.

The following settings are only available for job type "Execute index based evaluation":

- **Mandators**
  Select the mandators from which index based evaluations will be selectable.
- **Index based evaluations**
  Select the index based evaluations to execute. Only index based evaluations defined for the selected mandators will be available. Each index based evaluation will be executed isolated from the others so there is no interaction or dependency between them.

The following settings are only available for job type "export data job":

- **Export type**
  Type of exported data stream. Currently only CSV format is supported.
- **Target file**
  Directory and name of the file where exported data will be written. **It is highly recommended to use a different disk subsystem to avoid severe performance impacts.** You may include the following variable fields in the file name which will be replaced with actual values:
  - **{name}**
    This variable field is replaced with the name of the job.
  - **{comment}**
    This variable field is replaced with the comment of the job.
  - **{dateIso}**
    This variable field is replaced with execution date.
  - **{instanceId}**
    This variable field is replaced with the instance ID of the IBM Safer Payments instance.
  - **{instanceName}**
    This variable field is replaced with the instance name of the IBM Safer Payments instance.
- **Decimal separator**
  Specify which character should be used to as decimal separator.
- **Field separator**
  Specify which character should be used to seperate values in exported CSV file.
- **Salt**
  A random token which will be used to hash encrypted attributes. You can use the pre-generated salt, or you can enter a new one.
- **Include DDC**
  If enabled, the job will use data available on the disk data cache (DDC).
- **Export data selection**
  The data selection allows for both choosing an interval and additional conditions. Refer to the section help pages for more information.
- **Attributes**
  Allows to select which columns are to be included in the exported data file.
- **Encrypted attributes export**
  This section lets you define how encrypted attributes should be exported. Refer to the section help for more information.

Notice that when using multiple threads for data export, the data is processed in chunks and thus the sequence of records is not preserved. If you want to preserve the sequence you should execute single threaded data export.

back to top

8.11.2 Encrypted Attributes Export

When exporting encrypted attributes from IBM Safer Payments using data export job, you must define for every attribute how the stored value should be exported. Each encrypted attribute can be exported as plain text, and/or can be hashed or masked.

By default, plain text values of encrypted attributes are exported. However, you can deselect the plain text column, or add the hashed and masked variants of the value. Including hashed/masked variants of the value will introduce additional columns in the data export file named by *attribute name* + "_hashed" and *attribute name* + "_masked" respectively.

For exporting the hashed variant of an encrypted attribute, data export job uses so called salt, which is a random token used by the encryption algorithm to "safeguard" sensitive exports. You can use the pre-generated salt, or you can enter a new one in the field above.

back to top

8.11.3 Importing encrypted job files

If "Job encryption enabled" is activated in system settings in section "Batch data interface", encrypted job files can be imported through the BDI interface. To achieve the encrypted processing of batch data IBM Safer Payments chooses a combination of RSA and AES encryption. In order for the import to work, job files have to be encrypted with the following specification:

- Encryption system: AES (Advanced Encryption Standard)
- Cipher: aes-256-cbc
- Hash function: sha-256

It is expected, that job files may be encrypted by a different person than the one importing the encrypted files in IBM Safer Payments. Therefore encrypted job files and the AES password (which is used to encrypt the job files) need to be exchanged somehow. In order to exchange the AES password securely, IBM Safer Payments uses public / private key encryption using the RSA encryption system (Rivest, Shamir and Adleman).

Creating a RSA public an private key file

The person operating IBM Safer Payments has to create a private / public key pair. This can for example be done using openssl with the following command:

```
RSA_PASSWORD=myRsaPassword123

openssl genrsa 1024 | openssl pkcs8 -topk8 -out privateKey.pem -v1 PBE-SHA1-3DES -passout pass:$RSA_PASSWORD

openssl rsa -in privateKey.pem -out publicKey.pem -outform PEM -pubout -passin pass:$RSA_PASSWORD
```

A private key 'privateKey.pem' and a public key 'publicKey.pem' are then written to disk. The private key is secured with the "RSA_PASSWORD" and can only be used in combination with this password.

The private key stays at IBM Safer Payments and will be used in the following process to decrypt the AES password (see description below). The public key should be transferred to a person encrypting the job file.

Encrypting the job file

The person encrypting the job file can generate a 32 byte AES password (key) randomly with the following command.

```
AES_PASSWORD=$(< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c32)
```

The key does not have to be created randomly but can be defined manually, it is however recommended.

The AES password can be used to encrypt a job file with the following command:

```
openssl enc -aes-256-cbc -md sha256 -in unencryptedJobFile.csv -out encryptedJobFile.csv -pass pass:$AES_PASSWORD -p
```

The encrypted job file "encryptedJobFile.csv" is then written to disk.

Encrypting the AES password

With the RSA public key the unencrypted AES password can then be encrypted and written to disk with the following command:

```
printf $AES_PASSWORD | openssl rsautl -encrypt -inkey publicKey.pem -pubin -out aesKey.ssl
```

The AES password is then written encrypted to the file "aesKey.ssl"

Preparing the encrypted import

The person who encrypted the job file sends the IBM Safer Payments operating user

- the encrypted job file ("encryptedJobFile.csv" in the example, or multiple files if more than one were encrypted with the same password). The person operating IBM Safer Payments places this file in the "Incoming directory" folder which shall be used for the job .
- the file which contains the encrypted AES password ("aesKey.ssl" in the example). The person operating IBM Safer Payments places this file in a folder of the machine running the instance with the "Batch data interface".

The person operating IBM Safer Payments then places the previously created RSA private key inside thepassword safe folder (by default the folder "/pws") of each IBM Safer Payments instance. It is important, that the key is available on all instances, in order to activate it. The key must then be activated in 'administration/password safe' by selecting it in the table, providing the password RSA_PASSWORD from above and saving. Once the key is activated it will be marked with a green status symbol in the password safe table. See password safe for further details.

Importing the encrypted job file

To import the encrypted job file a job has to be created with "Load file job" as job type and the option "Enable encrypted import" needs to be activated. The password safe, which was activated in the previous step needs to be selected from the drop down in the field "Password safe". The path to the AES key file ("aesKey.ssl" in the example) needs to be provided in the field "Aes key path" and the path to the encrypted job file needs to be provided in the field "Incoming directory". Running the job then imports the encrypted job file. It is possible to import multiple encrypted job files, when these are placed inside the job file folder. Note that these files need to be encrypted with the same AES password in order for the decryption to succeed.

back to top

8.12 Password Safes

The table lists all password safes. A password safe is automatically generated for private keys which are located in the password safes folder.

Provide the correct password to activate a key. Once activated, it can be selected in an encrypted job to decrypt a file that contains an aes password. Using this aes password the encrypted job can be executed.

You will find further details for the encrypted job import in Importing encrypted job files.

back to top

8.12.1 Password Safe

Password safes have the following settings:

- **Name**
  Descriptive name. Can be selected in settings of an encrypted job.
- **Password**
  A private key itself should be protected with a password. Provide the correct password to activate the key. Once activated, it can be selected in an encrypted job to decrypt a file that contains an aes password. Using this aes password the encrypted job can be executed. You will find further details for the encrypted job import in Importing encrypted job files.
- **Comment**
  Used to describe the password safe.

back to top

8.13 Status Alarm Indicators

The table lists all status alarm indicators ("SAI"). SAI can be configured to monitor all important system and health parameter of an IBM Safer Payments installation (single-instance or clustered). They are displayed on the dashboard page and may be configured differently for each mandator.

back to top

8.13.1 Status Alarm Indicator

Status alarm indicators ("SAI") monitor internal or external parameter values of IBM Safer Payments operations. They can be defined as either "warnings" or "errors". If a defined threshold is reached, the SAI shows its warning/error status on the dashboard and (if enabled) sends out an email or mobile text message.

Configuration

The definition of a SAI involves a number of settings:

- **Enabled**
  Lets you enable/disable SAI without the need to deleting and re-entering them.
- **Position**
  Numerical value that identifies the relative position of the SAI on the dashboard. The smaller the number, the higher the position.
- **Name**
  Used to identify the SAI; can be taken into the SAI alert messages (below).
- **Comment**
  Used to describe the SAI; can be taken into the SAI alert messages (below).
- **Mandator**
  Each SAI strictly belongs to one mandator (this ownership cannot be changed once the SAI is created). You may choose from any SAI that you have SAI change privileges for. Notice that users will only see SAIs on their dashboard page that belong to the mandator the user belongs to as well.
- **Alarm type**
  Defines the type of alarm. Depending on the type of alarm, different subsections containing alarm type specific settings are shown. The alarm types are:
  - **Cases in investigation [#]**
    Monitors number of cases satisfying the specified settings:
    - Data from mandators
      Restricts the cases monitored to the mandators specified.
    - Case state
      Restricts the cases monitored to the case state specified.
    - Case close code
      Restricts the cases monitored to the case close codes specified.
    - Case conditions

Restricts the cases monitored to those that satisfy all conditions defined.

Notice that since cases are replicated within a cluster, this SAI will show only once on the dashboard.

- **Emails unsent [#]**
  Total number of emails and mobile text messages queued for sending out (total for all mandators defined in IBM Safer Payments). Since each IBM Safer Payments instance sends out its Emails and texts individually, this SAI will be shown for each cluster instance on the dashboard.
- **Externally delivered data**
  Reads SAI value from an external source. The additional setting is:
  - External file name
    Path and file name of the data delivered.

  For details on the file format, scroll to the end of this page.
- **Fastlink buffer usage [%]**
  Monitors utilization of the FastLink outgoing buffer. This SAI will be shown for each cluster instance on the dashboard.
- **Fastlink messages unsent [#]**
  Monitors backlog of the FastLink outgoing buffer. This SAI will be shown for each cluster instance on the dashboard.
- **IBM Safer Payments memory consumption [GB]**
  Current memory consumption of IBM Safer Payments in Gigabytes. This SAI will be shown for each cluster instance on the dashboard.
- **IBM Safer Payments reserved memory [%]**
  Current reserved memory of IBM Safer Payments in percentage of memory usage limit. Reserved Memory is the sum of simulation, analysis, rulegeneration and mdc memory of all champions. This SAI will be shown for each cluster instance on the dashboard.
- **IBM Safer Payments peak memory consumption [GB]**
  Peak memory consumption of IBM Safer Payments in Gigabytes. Same as before, only peak value as monitored by the operating system. Only available on Windows platforms. This SAI will be shown for each cluster instance on the dashboard.
- **Last error log message [h]**
  Time since last error log message was generated by this instance. This SAI will be shown for each cluster instance on the dashboard.
- **Last fatal error log message [h]**
  Time since last fatal error log message was generated by this instance. This SAI will be shown for each cluster instance on the dashboard.
- **Last warning log message [h]**
  Time since last warning log message was generated by this instance. This SAI will be shown for each cluster instance on the dashboard.
- **Encryption Key active since [d]**
  Time since current encryption keys was activated. This SAI will be shown for each cluster instance on the dashboard. Since encryption keys are replicated within a cluster, this SAI will show only once on the dashboard. *deprecated* use "encryption key valid until"
- **Remaining lifetime of key [d]**
  Time until current encryption key will expire and shutdown incoming interfaces. This SAI will be shown for each cluster instance on the dashboard. Since encryption keys are replicated within a cluster, this SAI will show only once on the dashboard.
- **Master Key active since [d]**
  Time since current master key will was activated. This function is deprecated and will be removed in a later release
- **Master Key valid until [d]**
  Time until current master key will expire and IBM Safer Payments will force a re-encryption. This SAI will be shown for each cluster instance on the dashboard. Since encryption keys are replicated within a cluster, this SAI will show only once on the dashboard.
- **Free disk space [GB]**
  The application will measure the number of available disk space in GB of this folder. On Linux for example, "statvfs" will be used to calculate the number of available blocks.
- **Oldest open case [d]**
  Monitors the age of the oldest unclosed case for the mandators defined. Since cases are replicated within a cluster, this SAI will show only once on the dashboard.
- **Index fill level [%]**
  Maximum fill level (calculated as the number of distinct entries currently stored in the index divided by the capacity of the respective index) of all indexes selected.
- **MCI average latency [ms]**
  Average internal latency in the selected period of time for messages in the Message Command Interface without network communication.
- **MCI maximum latency [ms]**
  Maximum internal latency measured in the selected period of time for messages in the Message Command Interface without network communication.
- **MCI latency violations [%]**
  Percentage of transactions that take longer to calculate than the maximum latency in settings.
- **Operating system total physical memory usage [%]**
  Percentage of physical RAM used on server. This includes IBM Safer Payments, the operating system, and all other software that runs on the server. If the available physical RAM is low, IBM Safer Payments operation may slow down to a degree that

renders stable operations impossible. Only available on Windows platforms. This SAI will be shown for each cluster instance on the dashboard.

    - **Transaction message rate [1/s]**
      Monitors number of transaction messages satisfying the specified settings:
        - Data from mandators
          Restricts the transaction messages monitored to the mandators specified.
        - Data points
          Lets you define a number of past periods used to average the value. Averaging is important with transaction message streams that have a high degree of fluctuation. The total period considered is printed for information right of the field (data points * check each).
        - Transaction message condition
          Lets you restrict transaction messages according to attribute values.

      This SAI will show for each cluster instance on the dashboard.
    - **Users logged on [#]**
      Number of users currently logged on.
        - Data from mandators
          Restricts the users monitored to the mandators specified.

      Notice that since users are served from only one instance within a cluster, this SAI will show only once on the dashboard.
- **Check each**
  Frequency this SAI is checked.
- **Alarm status**
  Each SAI can be defined to either indicate an "error" or a "warning". If you like both a warning and an error for an alarm status, define two SAIs with different thresholds.
- **Alert above/below**
  If "below" or "above" are checked, their respective threshold entry fields becomes visible and allow for the entry of values that if exceeded cause the SAI to go into alarm state
- **Dashboard messages**
  Specify how SAI are shown on the dashboard page:
    - **Display text**
      Template for text shown on the dashboard with each SAI line. Because of the limited space, you may want to keep this text short. You may use the message variables explained below to include dynamic contents.
    - **Display tooltip**
      Template for text shown as "tooltip" style pop-up message when the mouse pointer hovers over the display text. Because the tooltip is not hard limited on size, you may use this to display more details about the alarm than with the display text. You may use the message variables explained below to include dynamic contents.
- **Alert by Email**
  Enables SMTP delivery of email or text message. Once enabled, more fields are shown below:
    - **Email from**
      Email address (name@domain.com) that should be displayed as the sender of the alert. If this field is not set, the default email from address of the IBM Safer Payments settings is used.
    - **Email to**
      Email address (name@domain.com) of the recipient of the alert. Multiple recipients can be entered if separated by comma or semicolon.
    - **Email subject**
      Text template for the subject line of the email send out. You may use the message variables explained below to include dynamic contents.
    - **Email body**
      Text template for the email body send out. You may use the message variables explained below to include dynamic contents.

Template variables

Within the display text/tooltip and the email subject/body templates, you may use the following variable fields:

- **{externalMessage}**
  With all external alarms, the delivering file can contain a "message text". This variable field is replaced with that text. Notice that if this "message text" contains any of the other variable fields, they will be filled in as well.
- **{name}**
  This variable field is replaced with the name of the SAI. If the name text contains any of the other variable fields, they will be filled in as well.
- **{comment}**
  This variable field is replaced with the comment of the SAI. If the name text contains any of the other variable fields, they will be filled in as well.
- **{value}**
  This variable field is replaced with the current (numeric) value of the SAI using 3 decimals (when it was last computed).
- **{integerValue}**
  This variable field is replaced with the current (numeric) value of the SAI using no decimals (when it was last computed).
- **{thresholdAbove}**
  This variable field is replaced with the (numeric) value of the "above" threshold.

- **{thresholdBelow}**
  This variable field is replaced with the (numeric) value of the "below" threshold.
- **{lastUpdateTimestamp}**
  This variable field is replaced with timestamp the SAI was last computed (and the current value of {value} was derived).
- **{alarmTimestamp}**
  This variable field is replaced with timestamp the last alarm was triggered.
- **{mandator}**
  This variable field is replaced with the name of the mandator to which the SAI belongs.
- **{instanceId}**
  This variable field is replaced with the instance ID of the IBM Safer Payments instance.
- **{instanceName}**
  This variable field is replaced with the instance name of this IBM Safer Payments instance.

Template conditions

Before the template variables are filled in (above), you may define conditional text elements that are displayed only if the last computed value of the SAI meets defined criterion.

The format of these conditions is:
  IF(value *op value*:*text*)

As *op* (operators) you may use: "<", ">", "<=", ">=", and "=". *value* is the last computed value of this SAI and *text* is the text that is filled in instead of this condition when the condition is met.

For example:
  IF(value < 0.5:Last Warning {value}h ago)IF(value>=0.5:No warning within 0.5h)
will get replaced with the text "Last Warning 0.3h ago" if the value of the SAI was 0.3 and will get replaced with "No warning within 0.5h" if the value was 5.9.

Remarks

- All SAIs are computed periodically, each within its own IBM Safer Payments service thread.
- Each SAI computes/updates according to its frequency settings. If an alert (email/text) is generated, it is generated at exactly this time. The dashboard page refreshes according to its own refresh setting, or respectively, the user refresh actions, not necessarily synchronous to the SAI computational updates.
  With email generation, the email is generated when the alarm is first triggered. As long as the alarm remains active, the periodic computation does not trigger further alarms. Another alarm would only be generated if the alarm condition is first no longer valid, but then becomes valid again.
- There are no emails generated for the alarm condition being no longer valid.
- Explanation texts and online help options/texts to be displayed on the dashboard with SAI can be defined with the mandator administration settings.
- SAIs that monitor via a time series of data points are stored in memory only. After a restart of IBM Safer Payments, they must build up the time series before the reading is accurate.

External alarms

The alarm type "external" allows for external systems to deliver SAIs into the IBM Safer Payments dashboard and email alerting mechanisms. If alarm type is set to "external", a text entry "external file name" opens in which the path and file name of the file containing the external indicator values is defined. The file format is fixed-length text where each line corresponds to one value delivery. The "frequency" setting defines how often this file is read (last line only).

The format of each line is:
  YYYY-MM-DD hh:*mm*:*ss S mmm*

where *S* is the status (0: OK, 1: warning/error), and *mmm* the (variable size) message that optionally can be included with display and email messages as detailed above. The timestamp value is taken as the "alarmTimestamp" and can also be included in messages. Since IBM Safer Payments only reads the last line of the file, the delivering software program can safely append each alert to the file so that an audit trail gets created.

back to top

8.14 Charts

The table lists all charts that are defined and for which you have access privileges.

Charts are displayed on the IBM Safer Payments dashboard page and allow for key performance indicators ("KPIs") be defined within them.

back to top

8.14.1 Chart

Charts have the following settings:

- **Enabled**
  Lets you temporarily display or hide charts.
- **Position**
  Lets you define in which sequence multiple charts are shown. Smaller numbers are top positions.
- **Mandator**
  Each chart belongs to one mandator. Once created, mandator ownership does not change.
- **Name**
  Name that will be shown on the "dashboard" page of IBM Safer Payments.
- **Comment**
  Used to describe the case chart. The comment is also displayed to users on the "dashboard" page and may thus contain further explanations.
- **Explanation text**
  This text is shown below the header of this chart at all times.
- **Explanation tooltip**
  This text is shown as tooltip if the mouse pointer rests over the chart's header.
- **Online help type**
  Lets you enable default or custom help texts in the header of the status alarm section.
  - **Custom online help**
    If you selected custom help text to be displayed (above), you may enter it here.

back to top

8.15 Key Performance Indicators

The table lists all key performance indicators ("KPI") that are defined and that you have access privileges for. KPI are displayed in charts on the IBM Safer Payments dashboard page.

KPI are similar to status alarm indicators. Their differences are:

- KPI trace their indicators as time series and display them as time series charts on the IBM Safer Payments dashboard.
- KPI do not generate alarms or show alarm status. You may define both an SAI and a KPI for any indicator that you like both to be shown as time series and have (an) alert(s) associated to.
- KPI are also used to represent long-term data. They are thus stored both in main memory (for dashboard display) and on disk, so historical KPI data is automatically loaded when IBM Safer Payments boots.

back to top

8.15.1 Key Performance Indicator

The definition of a key performance indicator (KPI) involves a number of settings:

- **Enabled**
  Sets the KPI definition active. Notice that for each enabled KPI, a separate IBM Safer Payments service thread is started.
- **Chart**
  Selects in which chart this KPI is to be displayed.
- **Position**
  Numerical value that identifies the relative position of the KPI on the dashboard chart it is defined in. The smaller the number, the higher the position. The position values must not to be in sequence.
- **Name**
  This name is used to identify the KPI in the chart legend. The name should thus be kept short to not consume too much space. The chart legend displays for each time series: "*name* [*unit*/*timeunit*]" or "*name* [*unit*]".
- **Comment**
  This name is used to describe the KPI in the chart legend's tooltip.
- **KPI type**
  Defines the type of KPI. Depending on the type of KPI, different subsections containing KPI type specific settings are shown. The KPI types are:
  - **External file**
    External KPIs are read from disk to allow including data from other systems to be displayed with the IBM Safer Payments dashboard. Specific settings are:
    - External file name
      Path and file name of the data delivered.
    - Value unit
      Unit of the values of the data delivered (e.g. "trx/min", "cases", "%"). Will be printed as unit on the chart's diagram.

    For details on the file format, scroll to the end of this page.
  - **Investigation Activities**
    Monitors investigation activities that can be filtered with the following specific settings:
    - Data from mandators

Restricts the cases monitored to the mandators specified.

        - Case actions
          Restricts the cases monitored to selected investigation actions.
        - Case close codes
          If the case actions (above) to be monitored include closed cases, this selection opens and let you further restricts the closed cases monitored to the case close codes specified.
        - Case classes
          Restricts the cases monitored to the case classes specified.
        - Time unit
          Unit in which the KPI is computed and displayed.
        - Data points
          Lets you define a number of past periods used to average the value. Averaging is important with transaction message streams that have a high degree of fluctuation. The total period considered is printed for information right of the field (data points * check each).
        - Case conditions
          Restricts the cases monitored to those that satisfy all conditions defined.

    Notice that since cases are replicated within a cluster, this KPI will show only once on the dashboard.

    - **Logged on users**
      Number of users currently logged on.
        - Data from mandators
          Restricts the users monitored to the mandators specified.

      Notice that since users are served from only one instance within a cluster, this KPI will show only once on the dashboard.

    - **Notifications generated**
      Monitors number of notifications generated satisfying the specified settings:
        - Data from mandators
          Restricts the notifications monitored to the mandators specified.
        - Time unit
          Unit in which the KPI is displayed.
        - Data points
          Lets you define a number of past periods used to average the value. Averaging is important with notification streams that have a high degree of fluctuation. The total period considered is printed for information right of the field (data points * check each).
        - Transaction message condition
          Lets you restrict notifications according to attribute values of their transaction message.

      This KPI will show for each cluster instance on the dashboard.

    - **Number of cases**
      Counts number of cases by case classes and status.
        - Data from mandators
          Restricts the cases monitored to the mandators specified.
        - Case state
          Restricts the cases monitored to the case states specified.
        - Case close codes
          If the case state (above) to be monitored include closed cases, this selection opens and let you further restricts the closed cases monitored to the case close codes specified.
        - Case classes
          Restricts the cases monitored to the case classes specified.
        - Case conditions
          Restricts the cases monitored to those that satisfy all conditions defined.
    - **Average latency MCI**
      Monitors average latency of transaction messages in milliseconds.
    - **Maximum latency MCI**
      Monitors maximum latency of transaction messages in milliseconds.
    - **MCI latency violation**
      Monitors percentage of transactions that have required processing/latency of more than the defined threshold in configuration settings.
    - **Transaction message rate**
      Monitors number of transaction messages satisfying the specified settings:
        - Data from mandators
          Restricts the transaction messages monitored to the mandators specified.
        - Time unit
          Unit in which the KPI is computed and displayed.
        - Data points
          Lets you define a number of past periods used to average the value. Averaging is important with transaction message streams that have a high degree of fluctuation. The total period considered is printed for information right of the field (data points * check each).

        - Transaction message condition
          Lets you restrict transaction messages according to attribute values.

    This KPI will show for each cluster instance on the dashboard.

- **Frequency**
  Frequency this KPI is checked in seconds.
- **Period**
  Time period for which data points shall be kept in days.
- **Auto scale**
  If checked, the chart scales its own minimum and maximum according to the data range displayed. It not checked, you may enter a minimum and a maximum vertical axis value.
- **Representation**
  Lets you select which type of chart is to be used to visualize this KPI.

Remarks

- All KPIs are computed periodically within each its own IBM Safer Payments service thread.
- Each KPI updates according to its frequency settings. The dashboard page refreshes according to its own refresh setting, or the user refresh actions (whatever occurs first).
- KPI are stored in memory for the time period specified only. On disk, they are stored for an unlimited amount of time. After a reboot, past data is pre-loaded so that past performance data is shown as well.
- KPI data outside the defined time period is removed from memory once per day during the end of day gardening job (also for external KPI data files).

External alarms

The KPI type "external file" allows for external systems to deliver KPIs into the IBM Safer Payments dashboard. If KPI type is set to "external type", a text entry field "File name" opens in the form in which the file name of the file containing the external KPI is defined. Each external KPI corresponds to a text file named "kpi_*FileName*.iris" in the "KpiPath" directory. Each update cycle, the new contents of this file is read.

The format of each line is:

```
YYYY-MM-DD hh:mm:ss value
```

where *value* is the KPI value in floating point representation (for instance either "9.4150e+002" or "941.5"). The timestamp value is used to plot the time series data.

back to top

8.16 Messages

Transaction data enters and leaves IBM Safer Payments as "messages":

- online transaction messages (those that require an immediate response such as authorization requests) use IBM Safer Payments's message control interface (MCI), while
- offline transaction messages use IBM Safer Payments's batch data interface (BDI).

More information on these interfaces can be found at the Interfaces Overview.

In a typical IBM Safer Payments application, multiple data sources (and drains) exist that all send transaction message requests to IBM Safer Payments. Since these messages typically stem from different source systems, they typically contain different data fields that require mapping of message variables to IBM Safer Payments input and output attributes. They sometimes even contain different data formats and hence require (possibly different) preprocessing.

Messages and mappings

IBM Safer Payments comprises full management capabilities for messages and mappings. Because the message definitions themselves are model revision independent, messages are defined on a mandator basis within IBM Safer Payments administration. Messages are inherited downwards within the mandator hierarchy. Based on the own and inherited messages defined, within each model revision, the mapping of message variables to IBM Safer Payments attributes and any pre-/post-processing is defined.

The various data source messages are identified by a MessageTypeId (aka MTID), which is a (mandatory) IBM Safer Payments meta attribute of numeric data type. Typical message types in an IBM Safer Payments application could include: authorization requests, masterdata delivery transactions, posted transaction notifications, fraud alerts, chargeback notifications etc. Each different message can have its own variable-attribute mappings and its own pre-/post-processing settings.

back to top

8.16.1 Message

Each message is defined for one Message type ID. An incoming transaction message (via MCI or BDI) will be associated to a message if its Message type ID value fits the Message type ID.

The settings for each message are:

- **Mandator**
  Each message belongs to a mandator and is inherited to all mandators that belong to it. Notice that you cannot change mandator ownership of a message once created.
- **Name**
  Used in the model revision mappings to identify the message.
- **Comment**
  Used in the model revision mappings to describe the message.
- **MTID**
  Message type ID for this message as used with MCI transactions.
- **Storage and processing**
  Used to configure whether a message is creating transaction records.
  - *Create transaction records*
    Create a transaction record for this message. Can be overwritten by mergings and masterdata that are configured to store sources.
  - *Do not create transaction records*
    Do not create a transaction record for this message. Can be overwritten by mergings and masterdata that are configured to store sources.
  - *Compute monitoring lists only*
    Do not create a transaction record for this message and compute monitoring lists only. Cannot be overwritten.
- **Latency threshold**

  The maximum latency IBM Safer Payments may take for the processing of a transaction. This setting will be used by key performance, status alarm indicators and system internals.

- **Monitor latency**
  Monitor latency as key performance indicator or in latency report, if available. Only messages whith this checkbox enabled will be considered as latency violation for KPIs and will be added to the latency report in system internals. The latency report is archived in the "log" directory and can be downloaded with the configuration in "System Internals" page.
- **Offline via Batch Data Interface**
- **Type**
  Used to identify if a message is used online (MCI) or offline (BDI). In case of offline messages the message can either be a CSV, FCD, JSON, XML or nested XML file. The MCI interface accepts XML, JSON and custom defined messages. For custom messages, please make sure you have the custom parser library in place.
- **Charging message**
  Used to configure whether transactions of this message are to be charged. Only charging messages will be counted in the transaction message report.
- **Fixed entry size**
  Used to determine how many characters need to be read from file per line/entry. If -1, then no fixed sized will be applied, and it will be read till end of line.
- **Online via Message Command Interface**
  IBM Safer Payments uses these values to estimate bandwidth required for replicating and processing of MCI data. Please enter the expected values in the respective fields.
  - *Average volume*
    Total number of transaction messages within a long time period (typically a year).
  - *Peak volume*
    Peak number of transaction messages to be processed (typically defined as per second).
- **Offline via Batch Data Interface**
  IBM Safer Payments uses these values to estimate bandwidth required for replicating and processing of BDI data. Please enter the expected values in the respective fields.
  - *Records per file*
    Number of records on average per file.
  - *Volume*
    Number of files delivered per time unit.
  - *Processing time*
    Time in minutes one of the delivered files shall be processed within.

back to top

8.16.2 Messages Report

This report shows an overview of all messages' data streams.
back to top

8.16.2.1 Message Report

The message report provides aggregated and computed information on a message type. Each message type can come into IBM Safer Payments as online and/or offline data streams. Open the respective help pages of the subsection(s) for more detail.

This information is intended to provide assistance with IBM Safer Payments sizing and the implementation of interfaces to IBM Safer Payments.

back to top

8.16.2.1.1 Online via Message Command Interface

For messages that are defined to receive online transaction messages via MCI, the estimated lengths of the messages is shown plus example messages (with variable formatting and additional information). In addition, bandwidth requirements are computed from the volume settings provided from the message definition. The MCI interface accepts the XML or JSON format by default. In case you want to send other formats, please configure the MCI settings accordingly and make sure you have the custom parser library in place.

Message samples

The samples provided illustrate how messages to and from IBM Safer Payments should look like. Notice that there are a number of placeholders used:

- The MessageId value is exemplary.
- 'xxxx' denotes hexadecimal values (number of characters represents maximum length).
- 'aaaa' denotes text values (number of characters represents maximum length).
- '1' denotes Boolean values.
- Numeric values have placeholders showing the maximum positive number that the IBM Safer Payments attribute can take.
- IPv4 values are denoted by an exemplary IP address.
- '....' is a placeholder for value of unknown type (for instance, if used by a processing function).
- 'YYYY-MM-DD hh:mm:ss' is a placeholder for a timestamp value (depending on processings, there could be different formats).

Refer to the reference on the MCI (Message and Command Interface) for a complete reference on messages sent to and received from IBM Safer Payments.

Message samples view options

The different view options to the message sample are:

- condensed
  If the placeholders are replaced with the true values, this is *exactly* the format of the message IBM Safer Payments expects to receive or will send. Since this format does now allow for line feeds, the example can be rather long and you might have to scroll the page to the right.
- pretty
  Different to 'condensed', this option uses line feeds and space characters to make the sample message easier to read.
- pretty plus warnings (only XML)
  Also shows when different IBM Safer Payments attributes feed from the same XML variable.
- pretty plus warnings and comments (only XML)
  Provides additional information on which IBM Safer Payments attributes feed from the XML variables.

back to top

8.16.2.1.2 Offline via Batch Data Interface

For messages that are defined to be delivered with batch file data, the estimated length of records is computed. In addition, bandwidth requirements are computed from the volume settings provided from the message definition.

FCD columns format

This section provides an exact overview on what data all mandators extract from an FCD file, in case the FCD format is enabled for this message.

For each data element extracted, the exact positions of extraction are listed and the attributes that feed from it. The color icon indicates potential warnings or errors with the definition. Positions within the FCD record that are not extracted are also listed.

FCD record sample

This section provides an example record using the following placeholders:

- Boolean
  A Boolean value is indicated by the '0' placeholder character.
- Numeric
  A numeric value is indicated by '1' placeholder character(s).
- Hexadecimal
  A hexadecimal value is indicated by 'x' placeholder character(s).
- IPv4
  An IP address value is indicated by 'i' placeholder character(s).
- Text
  A text value is indicated by 'a' placeholder character(s).
- Timestamp

A timestamp value is indicated by its complete format placeholder (e.g. 'YYYYMMDD').

Notice that positions within the FCD record that are not extracted by any mandator's mapping are indicated by '_' characters, and positions extracted to multiple attributes are indicated as '?'.

back to top

8.17 Real-time Intercept Codes

The table lists all real-time intercept codes ("RIC") that are defined and for which you have access privileges.

Real-time intercept codes are the values of the "intercept" meta attribute that control the real-time decision IBM Safer Payments feeds back. Typically the authorisation system that receives this value carries out the action recommended by IBM Safer Payments

back to top

8.17.1 Real-time Intercept Code

Real-time intercept codes have the following settings:

- **Name**
  Descriptive name as used for IBM Safer Payments reports.
- **Comment**
  Used to describe the real-time intercept code. The comment is also displayed to users on selected reports.
- **Interval**
  Defines the value(s) for the meta attribute "intercept" that fall into this real-time intercept code. Notice that if the real-time intercept code is only one value, you mayenter only this value both as "from" and "to" as the interval definition is inclusive.
- **Intercept**
  There are three different types of intercepts:
  - **authorise**
    IBM Safer Payments has no objection to authorise this transaction.
  - **refer**
    IBM Safer Payments recommends to refer this transaction to an investigator.
  - **decline**
    IBM Safer Payments recommends to decline this transaction.

  Note that all transactions with an intercept code 'refer' or 'decline' are considered as 'marked intercepted' within analyses.

back to top

8.18 Event Logging

This sections covers IBM Safer Payments event logging.

back to top

8.18.1 Configuration

This sections covers event message configuration.

back to top

8.18.1.1 Event Log Messages

IBM Safer Payments comes with a fully configurable event log engine that is configured from the event log message page on the admin tab. The page lists all available event log messages.

The log messages are programmed into IBM Safer Payments, you may thus not add or delete from this list. Click on a row to view details and to change settings.

Notice that all settings that are user overwritten are indicated with an asterisk behind the value.

back to top

8.18.1.2 Event Log Message

This section shows settings of the log message identified in the section title:

- **PCI DSS mandated**
  To comply with PCI DSS, certain log messages must be included in the audit logs. This is indicated for each log message and cannot be changed by the user. However, this indication is not mandatory and should be considered a recommendation. You may choose to follow this recommendation or not.
- **Default**
  These are the settings as defined by the IBM Safer Payments team as default values. You may override these settings in the "Change Settings" section below.
- **Change Settings**
  Each default setting (above) may be overwritten by first checking the "Change xyz" checkbox, and then entering the new value for the setting. Notice that the changing remains even if IBM changes a setting in the future. For example, if log message X be disabled and you change it to enabled, the log message is shown. If later, IBM decides to enable the log message by default, the log message is still shown. If even more later, IBM changes the default again to disabled, the log message will still be shown. However, once you uncheck the "Change xyz" checkbox, the default setting is used.

Log targets

Log target configuration allows enabling/disabling the event log message for the three possible targets:

- **System event log**
- **Audit event log**
- **Operating system event log**

System and audit event logs are IBM Safer Payments internal. The log messages are stored in daily files in the "log" directory of the IBM Safer Payments installation and can be read from the respective pages on the admin tab. Alternatively, the simple text format of the log message files also allows them to be read with any editor or processed with any log tool.

The operating system event log target stands for the external event logging system of the operating system, where Safer Payments is running on. On UNIX operating systems, event log messages are delivered as "syslog" messages.

Log levels

The log level is a classification involving the levels:

- **Debug**
  Analysis and troubleshooting information mostly used during integration and testing of an IBM Safer Payments installation.
- **Copyright**
  Delivers release information.
- **Informational**
  Describes certain activities within IBM Safer Payments.
- **Warning**
  Potentially troublesome incidents.
- **Error**
  Problems presumably resolvable by user/administrator.
- **Fatal**
  Problems presumably resolvable only by IBM. If you encounter them, please contact IBM Safer Payments support to analyze.

The comment of the log message is displayed only in log tables. It gives additional information about the general circumstances of a log event and how to resolve certain situations. Please notice that this is not the actual message that is printed to the system or audit event logs.

back to top

8.18.2 Browsing

This sections covers event message browsing.

back to top

8.18.2.1 Log Entries

This page lets you view log messages:

- Which of the generated log messages are displayed depends on the filter settings above.
- How and which log messages are generated depends on the event log message settings.
- The columns displayed are:
  - **Instance**
    IBM Safer Payments optionally (system configuration) implements centralized logging. In this case, all IBM Safer Payments instances replicate all locally generated log messages to all other instances within the IBM Safer Payments cluster. This column identifies the IBM Safer Payments cluster instance this log message was originated on.
  - **Timestamp**
    Timestamp log message was generated (system time of originating instance), including microseconds. Notice that the actual resolution of the computer clock might not represent microseconds.
  - **Log level**

    The log level is a classification involving the levels:
        - **Debug**
          Analysis and troubleshooting information mostly used during integration and testing of an IBM Safer Payments installation.
        - **Copyright**
          Delivers release information.
        - **Informational**
          Describes certain activities within IBM Safer Payments.
        - **Warning**
          Potentially troublesome incidents.
        - **Error**
          Problems presumably resolvable by user/administrator.
        - **Fatal**
          Problems presumably resolvable only by IBM. If you encounter them, please contact IBM Safer Payments support to analyze.
    - **ID**
      The log message ID is a unique identification within IBM Safer Payments. Notice that the same log message (same ID) can have different log message texts depending on what is intended to express.
    - **User**
      For all log messages that are generated with respect to an explicit action of a user, the user login (and name) is provided here.
    - **Message**
      Text describing the actual event. Notice that with the audit log, the IP address from which a request is received is included after message text. For requests that were forwarded to IBM Safer Payments by a (reverse) proxy, the originating IP address(es) are also provided.
    - **Comment**
      This text provides further explanation of the log message and may contain information on how to provide remidy for a problem.

  Notice that you may change log level and comment of each log message from the event log message page.

back to top

8.18.2.2 Log Filter

The settings of this section enable filtering of log entries displayed in the section below. A log entry must satisfy all filter conditions to be displayed. The filter conditions are:

- **Date/time**
  Lets you define a time interval.
  By default this is defined as the time period from now into the past using the number of seconds defined in system configuration setting 'Event log messages - Default view period'.
- **Log levels**
  Lets you select which log levels should be displayed.
- **Instances**
  In a multi-instance cluster installation with centralized logging enabled, you may select instances for which log entries shall be shown.
- **Users**
  Select only log messages that are associated to a specific user. Notice that "[IBM Safer Payments]" is a placeholder for log messages that are not associated to a user.

All changes to filter criteria are executed immediately, that is, as soon as the entry focus leaves an entry field, the contents of the log entries table below refreshes according to the new filter criteria.

The button **Refresh table with default view period** (new user interface only) will refresh the table with the default timespan mentioned under 'Date/time' above.

back to top

8.19 Memory Control

Certain modelling simulations of model revisions require IBM Safer Payments to create temporary data objects. For instance, if you define a new or modified counter in a model revision and enable simulation for it, IBM Safer Payments must create temporary data objects to represent the output attributes of the counter. Since simulation is mostly performed out of main memory (rather than disk memory) for performance reasons, and since main memory is a scarce resource, once the main memory available to simulations is exhausted, IBM Safer Payments has to decline further requests for simulation.

The memory control page first provides an overview on the memory consumption of all simulations running and the information when this model revision was last touched by the user. Second, users with respective privileges may stop simulations of users to free their memory for other simulations. This ensures that there are no "forgotten" simulations that consume the otherwise dearly needed memory.

To create a partial list of simulations, you may filter the table for both mandators and users.

back to top

8.20 System Configuration

This page contains all settings that are the same for all IBM Safer Payments instances in a cluster. Their settings are described by tooltip style explanations when you rest the mouse pointer over the respective label.

Typically these settings have been made during installation of IBM Safer Payments by or with the assistance of IBM Safer Payments consultants. Only perform changes on this page when you know what you are doing as improper settings can cause IBM Safer Payments to not work as excepted.

Notice that some changes will come into effect immediately, while others come into effect only when IBM Safer Payments reboots. This can cause different IBM Safer Payments instance in a cluster to behave differently when re-booted at different times.

Because the subjects of the different sections of this page are highly divers, detailed help is provided from within each section for some of the sections.

When in doubt, contact the IBM Safer Payments support to assist with the configuration.

Tools

The toolbar of this section contains some utilities:

- Saves the system configuration (replicated via FLI to all instances of the cluster).
- Lets you download an archive file with configuration and log data to your local computer (to be used with issue analysis).
- Creates a full report with respect to the PCI DSS compliance of the current IBM Safer Payments system configuration and settings.

back to top

8.20.1 Main Memory Sizing

IBM Safer Payments draws part of its high transaction message processing, simulation, and model generation performance from the fact, that it caches most of the data used for computation in main memory. IBM Safer Payments uses main memory for different purposes:

- Production memory data cache
  For each data holding attribute, index, calendar profile, etc in IBM Safer Payments, the part most relevant to real-time processing (aka "production") is kept in this cache area for expedited access. In most IBM Safer Payments applications, the total main memory requirement of the production memory data caches requires the dominant part of the main memory available on the server hardware. How much memory is used depends on the individual settings of all champions' model elements. For details, refer to IBM Safer Payments storage architecture.
- Simulations memory data cache
  During simulation, IBM Safer Payments needs to build up temporary memory data caches for simulated elements. For instance, if a new element was added or changed, and its result is to be simulated, IBM Safer Payments uses caching so that the results are immediately available for analyses and model generation.
- Miscellaneous
  While most other functions of IBM Safer Payments also use main memory during computation, this amount is significantly smaller compared to the MDC memory used for production and simulation.

Notice that the operating system will deliver IBM Safer Payments with nearly any amount of memory, yet once physical RAM memory is exhausted, the operating system will start offload RAM pages to disk (swapping). This diminishes IBM Safer Payments performance by multiple orders of magnitude to a level where the entire computer can become so busy with swapping, that it does not react anymore to user input or transaction messages.

It is obvious that this situation must be avoided at all cost. IBM Safer Payments has a number of features to configure the amount of main memory accessible to data caches and simulation as well as means to help monitor the key health figures of the installation:

- Sizing
  - Main memory usage limit
    The maximum memory consumption of any IBM Safer Payments instances can be limited.
  - Mandator simulation memory limit
    For each mandator the maximum of simulation memory that can be accessed for analyses, simulations and model generation can be limited.
- Alarming
  - Status Alarms Indicators
    IBM Safer Payments provides various Status Alarm Indicators to monitor the memory consumption of an IBM Safer Payments installation. The monitoring is performed on instance level an can be accumulated per cluster. For details, refer to Status Alarm Indicators.

back to top

8.20.2 Authentication Settings

IBM Safer Payments supports the following authentication methods:

- Local
  Only the IBM Safer Payments stored password is checked.
- LDAP
  Only the LDAP password is checked.
- Local and LDAP
  First the local password is checked, if it does not fit the LDAP server is checked.
- OIDC claim
  The specified username attribute inside the OIDC token is checked for existance within IBM Safer Payments

LDAP authentication

IBM Safer Payments supports LDAP (lightweight directory access protocol) servers as an alternative means of user access authentication. The LDAP server is not part of IBM Safer Payments, you must install and operate it separately from IBM Safer Payments.

Configuration of the LDAP server involves the following settings:

• Use Active Directory
Use Active Directory authentication method (username@domain)

• Allow Single Sign On
Allow a user to automatically login using their windows credentials. For more info please see Setup required to enable Single Sign On.

• Automatically Re-login
Allow a user to automatically sign in after their session expires (this setting is only available when Single Sign On is enabled).

• LDAP hosts
IP address or domain name of LDAP server (for example "iris.ldap.intranet"). You can also enter multiple hosts by separating them with commas (i.e. "iris.ldap.intranet,iris2.ldap.intranet"). In this case IBM Safer Payments will try and connect to any backup hosts in the order they are listed if it cannot contact the first host.

• LDAP base DN
Base of distinguished name (for example "OU=Users,DC=company,DC=local").

• Active Directory domain
Active Directory domain for login (username@domain).

• LDAP port
IP port of LDAP server (typically "389" or "636" for SSL access).

• LDAP encryption over SSL
Enables encrypted communication between IBM Safer Payments and LDAP host.

The LDAP SSL connection settings are configured outside of Safer Payments.

Safer Payments is using the shared library of OpenLDAP to implement the LDAP authentication. The SSL/TLS settings are not done in Safer Payments, they need to be applied to the OpenLDAP configuration file, which can be found under /etc/openldap/ldap.conf. Please refer to the OpenLDAP manual for the necessary settings and options to set up the certificates. Please note that you have to restart Safer Payments after every change of configuration files on operating system level for the changes to take effect in Safer Payments.

• LDAP timeout
LDAP timeout (time to wait to connect to LDAP server)

Notice that in each case, a user account in IBM Safer Payments must exist for a user to log on.

If LDAP only authentication is enabled, all administrators are also authenticated only by the LDAP host. If this authentication fails, the administrator is locked out.

OIDC authentication

IBM Safer Paymets also supports a token based Single Sign On (SSO) login based on the OpenID Connect protocol. In this case, Safer Payments is presented with a OIDC token containing a login name. This token is provided by an external server, which is not a part of IBM Safer Payments and must be operated separately. For PCI compliant usage, you must address all PA-DSS and PCI DSS requirements for this server on you own.

The configuration of the OIDC token based authentication involves the following settings:

- Token name
  In this field, you need to specify the name of the HTTP header field that contains the OIDC claims token.
- Username attribute
  The username attribute identifies the attribute of the OIDC claim that is used for authentication.

Example token:

```
{
"login_name": "jsmith@exampleMail.com",
"last_name": "Smith",
"first_name": "Jane",
"id": "cl.jsmith",
"internal_id": "25",
"emails": {
```

```
"personal": "jsmith@personalMail.com",
"business": "jsmith@businessMail.com"
},
"phone_numbers": {
"business": "555-555-1234"
}
}
```

If this login method is used, you must ensure that the username submitted via the OIDC claim token exists as a user inside IBM Safer Payments. If this is not the case, a login is not possible.

back to top

8.20.2.1 Setup required to enable Single Sign On

When you have configured Safer Payments to use LDAP (with either the 'LDAP' setting or 'local then LDAP') you also have the option to turn on Single Sign On (SSO) allowing your users to be automatically logged in by authenticating the account that they've logged in with on their machine. In order to do this the Safer Payments server uses the Kerberos authentication method and some additional set up steps will be required on your LDAP server, on the Safer Payments server and on each individual's client machine. The first steps will describe how to set up SSO assuming you are using Active Directory as an LDAP server. If you are using a Linux KDC then see the steps at the bottom for instructions.

The steps below need to be performed for the Safer Payments server where your API is running. If you would like SSO to be available if your API changes servers then you will need to perform the steps on each other server where the API can run. The steps below will assume that you are just setting up kerberos on the single Safer Payments server where the API is running, and so will refer to this server as the Safer Payments server

Creating the Kerberos keytab on Active Directory

First, go to your Active Directory server and add a new user which will be used by the service (if you are using a Linux KDC see the bottom of this document for instructions). In our examples below we will assume that the user is named *SPServiceUser* with the password *Password123*. Ensure that this user's password will not expire, since it will only be used by the service. We will also assume that you have another user that will be used to login to Safer Payments called AdminUser with a password of your choosing.

Next open a command prompt as administrator and run the following 2 commands to associate the service principal with the username and create the keytab:

```
setspn -A HTTP/SaferPaymentsServerWithDomain Username

ktpass -out OutputPath -princ HTTP/SaferPaymentsServerWithDomain@DomainNameInCaps -mapUser Username -mapOp set
-pass Password -crypto RC4-HMAC-NT -pType KRB5_NT_PRINCIPAL
```

For example, if your Safer Payments server is on a machine called *SPServer* and is part of the domain *internal.example.com* and you used the username and password mentioned previously you would run the following commands

```
setspn -A HTTP/SPServer.internal.example.com SPServiceUser

ktpass -out c:\krb5.keytab -princ HTTP/SPServer.internal.example.com@INTERNAL.EXAMPLE.COM -mapUser SPServiceUser
-mapOp set -pass Password123 -crypto RC4-HMAC-NT -pType KRB5_NT_PRINCIPAL
```

You should now have a keytab file created at *c:\krb5.keytab*. Copy this file to your Safer Payments server where your API is running using a secure method.

Configuring Kerberos on the Safer Payments Server

Next you will do the required Kerberos configuration steps on the Safer Payments Server. First, you want to ensure that you can reach the Active Directory server from your Safer Payments server with the fully qualified domain name (i.e. assuming the Active Directory server is named *ADServer* and continuing with the example above you would run *ping ADServer.internal.example.com*). If you can't reach the Active Directory server then you'll need to either configure your hosts file or your local DNS server so that you can.

After ensuring that your Safer Payments server can connect to your Active Directory server you will set up the */etc/krb5.conf* file to point to your domain. Open the /etc/krb5.conf file in your editor and change the default_realm to point to your realm, and then change the [realms] section to point to your Active Directory server and the [domain_realm] section to point to your realm. For example, continuing with the example above the file would look like this (all the logging and libdefaults settings other than the default_realm can potentially be changed).

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = INTERNAL.EXAMPLE.COM
```

```
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true

[realms]
INTERNAL.EXAMPLE.COM = {
kdc = ADServer.internal.example.com
admin_server = ADServer.internal.example.com
}

[domain_realm]
.internal.example.com = INTERNAL.EXAMPLE.COM
internal.example.com = INTERNAL.EXAMPLE.COM
```

You can now test that your kerberos settings are correct and you can connect to your Active Directory by running the *kinit* command. Continuing with the example above you would run the command

```
kinit AdminUser
```

Followed by the user's password. If no errors occur then you can run the *klist* command which should show a ticket for the username which you just entered. If you had an error while running the kinit command, check the connectivity between the machines (in both directions) and ensure that there were no errors in either of the configuration files you set up in the previous steps

Next copy the keytab file you created from the previous section to /etc/krb5.keytab

Since keytab files contain highly sensitive information, notably encryption keys, it is imperative to ensure proper access controls to these files. Assuming that your Safer Payments service runs under a unique username, the keytab file should be modified so it is readable only by that username.

You can now test that the keytab was created correctly. Run the following command and ensure that it lists the service principal which you created in the steps above.

```
klist -k -t /etc/krb5.keytab
```

With our example above, the command should show the principal HTTP/SPServer.internal.example.com@INTERNAL.EXAMPLE.COM

Next you can test that you can authenticate the keytab by running the following command (followed by entering the user's password that you set up previously):

```
kinit -k -t /etc/krb5.keytab HTTP/SaferPaymentsServerWithDomain
```

For example, continuing with the example above you would enter:

```
kinit -k -t /etc/krb5.keytab HTTP/SPServer.internal.example.com
```

Followed by the password *Password123*. If this command does not give any errors then your keytab is set up correctly and kerberos is correctly configured on the machine.

Setting up Safer Payments

In order to use the Single Sign On feature you need select the 'LDAP' or 'local then LDAP' settings in the system configuration. Once you have selected one of those options you will have the option to 'Allow Single Sign On' which will allow the users to be automatically logged in with the same user account that they used to log in to their machine when they load any Safer Payments page. The Single Sign On feature assumes that the usernames are the exact same between Safer Payments and the Active Directory server, just like the LDAP feature.

Setting up the client

Finally, before a client can use the Single Sign On feature you will need to run some set up steps on your client machine to allow your authorization credentials to be sent through the browser.

First, obviously you will need to ensure that the machine is connected to the domain and that you are logged in as a domain user who also has a username in Safer Payments

Next you will need to set up the browser. Follow the steps below depending on which browser you are using.

Internet Explorer and Google Chrome on Windows

1. Open *Control Panel* and select *Internet Options* and then the *Security* tab.
2. Select the *Trusted sites* zone and then select the *Custom level...* button
3. In the settings dialog navigate to the *User Authentication* section and change the *Logon* setting to *Automatic logon with current user name and password* and hit OK

4. Finally, back on the *Trusted sites* zone, hit the *Sites* button and add the website of your Safer Payments instance to the Trusted sites (using its fully qualified domain name). Following with our previous example we would add http://SPServer.internal.example.com (if your Safer Payments site only uses HTTP and not HTTPS you will need to unselect the box at the bottom requiring HTTPS)
5. You can now hit ok to close the settings dialog and save the settings

Firefox

1. Open a new Firefox tab and in the URL bar navigate to *about:config* and click the *I'll be careful, I Promise!* button
2. In the search dialog search for *negotiate* and then open the *network.negotiate-auth.trusted-uris* value and enter the fully qualified domain name (following with our previous example you would enter SPServer.internal.example.com)
3. You can now close the *about:config* tab

After configuring the browser, ensure you are logged on to your machine with a user account that has a matching username in Safer Payments, and then you should be able to open a new web page pointing to Safer Payments and be automatically logged in.

Creating the Kerberos keytab on Linux

The information below will describe how to create a keytab using the MIT KDC implementation, however there should be equivalent commands in other implementations.

Ideally you should set up your Iris server to access the KDC before running the steps below, so that you can run the remote version of kadmin on your Iris server. To do that, follow the steps above under *Configuring Kerberos on the Safer Payments Server* until after you run the *kinit AdminUser* command. Alternatively, you can also run the kadmin.local command on your KDC server, but it is recommended to run it on the Iris server so you do not need to transfer the keytab over the network.

First you will need to create a principal for the service. Run the kadmin command to enter the kadmin CLI, then run the following commands to add a service principal and then create a keytab.

```
kadmin

addprinc HTTP/SaferPaymentsServerWithDomainInLowercase[@RealmNameInCaps]

ktadd -k OutputPath HTTP/SaferPaymentsServerWithDomainInLowercase[@RealmNameInCaps]
```

To use our example names from above you would do the following commands:

```
kadmin

addprinc HTTP/spserver.internal.example.com@INTERNAL.EXAMPLE.COM

ktadd -k /home/dev/KDCServer.keytab HTTP/spserver.internal.example.com@INTERNAL.EXAMPLE.COM
```

You will now have the keytab file available on the Iris server (or, if you ran kadmin.local on the KDC machine, transfer the keytab file to the Iris server using a secure method) and you can follow the remaining steps from *Configuring Kerberos on the Safer Payments Server* above
back to top

8.20.3 Deferred Writing

The IBM Safer Payments storage architecture involves both an in-memory data cache layer (MDC) and a disk data cache layer (DDC). The standard IBM Safer Payments data access logic is the following:

- Read access
  For any kind of computation need, transaction data available in MDC is read from MDC. Data not available in MDC is optionally read from DDC (the latter should be used not in real-time decisioning, such as in profiling counters or mergings, and its use can be enabled or disabled in the system configuration).
- Write access
  Data is both written to the MDC and the DDC immediately (if it falls within the limits of the respective storage capacity).

High performance applications

For high-performance applications of IBM Safer Payments, with a very large number of online or offline transactions to be processed (multiple thousands per second), the immediate (synchronous) writing of each data element into the DDC can be "deferred". This way, multiple DDC write operations are combined (so that they can be written to disk more efficiently) and are written a short time later (asynchronous).

If deferred writing is enabled, IBM Safer Payments does only write data (that the MDC can hold) into the MDC (and not the DDC immediately). Rather a separate service thread continuously stores all unsaved data (data in MDC but not in DDC) in a round robin fashion. That is, first all unsaved records for each attribute are saved at once, then all indexes.

It should be noted that even though this approach of storing IBM Safer Payments data on disk in the end transfers the same total amount of data compared to the IBM Safer Payments standard data access logic that stores each record's data individually and immediately, however, since storing larger chunks of data rather than many many small elements is by orders of magnitude more efficient.

Deferred writing thus enables using IBM Safer Payments in high performance applications while it ensures that data loss is minimal in case of a non-orderly shutdown.

Deferred writing options

Optionally a deferred writing time gap can be defined. If this period is set to larger than zero, this time period will never be saved to disk. This option is useful for operations of IBM Safer Payments in applications where the actual decision on whether or not to authorized a transaction is made after IBM Safer Payments has assessed the transaction, and IBM Safer Payments is informed of the (final) authorisation decision by advices. Since such advices messages would come into IBM Safer Payments with a certain delay (usually seconds), deferred writing should wait a period somewhat larger than the advice delay, so that the merging operation can be performed while the data is still in main memory. Notice that if the advice for whatever reason comes later and the respective data range has already been written to disk, the merging is still carried out correctly, yet because the merging target writes are into an region where the main memory attributes are already stored on disk, the merging targets would induce direct disk writes, that can severely slow down transaction processing.

Another deferred writing option is the setting of a safety margin. Since IBM Safer Payments continues to process transactions while the data from main memory is written to disk, not the full MDC cache size is available for deferred write caching. This setting defines how many MDC record positions are *not* used for deferred writing caching. The value should be set as the maximum number of transactions (messages and records) processed by IBM Safer Payments in the time it takes to write the MDC cached data to DDC.

Deferred writing disk usage limits

Depending on the performance of the disk subsystem, IBM Safer Payments can overuse caches and control mechanisms. It is therefore useful to limit the bandwidth of IBM Safer Payments deferred writing operation. This limit is facilitated by writing data in chunks of 'disk chunk size' and wait the 'disk chunk delay' amount of time after each chunk was written.

For most purposes, the optimum chunk size is 64 KB (65,536 Bytes). The disk chunk delay can then be computed by the following formula:

```
disk chunk delay = ((1/SET) - (1/NET)) * disk chunk size
```

where

- SET is the desired maximum write performance, and
- NET is the maximum sustained writer performance of the disk subsystem.

Assuming a NET performance of 128 MB/s and a desired IBM Safer Payments write performance of 48 MB/s, as well as a disk chunk size of 64 KB, the disk chunk delay would compute to about 1 millisecond.

Notice that for technical reasons, the minimum wait time is 1 milliseconds, if the write performance should be higher, the disk chunk size would have to be increased from 64 KB.

Since data elements smaller than the defined disk chunk size are written at once, it can be useful to define a pause after IBM Safer Payments has written an element (e.g. an attribute, an index, a masterdata) to provide the disks some time to perform other tasks. The typical setting for the 'pause after element' is the same wait time as the disk chunk delay.

Deferred writing shadow commit options

There are currently two options for the deferred writing shadow commit:

- shadow commit chunk size
  If greater than 0, transaction computation will get smoother as between chunks, computation continues.
- shadow commit chunk delay
  Time shadow commit waits at least between chunks for transaction computation (if greater than zero).

Remarks

- If deferred writing is enabled, IBM Safer Payments starts a separate service thread that continuously subsequently writes all attributes, all indexes, all masterdatas, all events, and all calendar profiles. This implies that depending on the wait times for the disk subsystem, this thread alone can pretty much use the computing resources of an entire CPU. You will thus observe CPU load on IBM Safer Payments even if IBM Safer Payments does not perform any other task.
- While with attributes, only the period (record interval) between the last MDC to DDC save operation is stored, indexes are stored at once. Since the actual MDC to DDC save operation takes too much time for transaction processing to wait for its completion, write operations to the index/masterdata/event/calendar profile are cached in a temporary MDC (aka "shadow") during the MDC to DDC save operation. Once this save operation is completed, the temporary MDC will be committed at once to the respective index/masterdata/event/calendar profile (to the MDC, where it is saved in the next deferred write iteration).
- The time it takes to have written all attributes' deltas and indexes depends on the number of transactions IBM Safer Payments processes and the performance of the hardware used; in particular its disk subsystem performance. This is a self-regulating process as the higher the transaction volume gets, the larger the stored chunks of data get (and thus the efficiency of the save operation increases. It is not possible to deliver more transactions to IBM Safer Payments as the deferred writer can save in a typical hardware configuration.
- When IBM Safer Payments stops to compute transactions (offline and online), the to be cached record intervals for the attributes eventually reaches zero and the indexes have no changes. An orderly shutdown will also ensure that all cached data is stored from MDC to DDC before IBM Safer Payments stops operation.
- Enabling deferred writing in operation will not cause any disruption of service as the individual data element writing is just disabled and the asynchronous deferred writing service thread is started.
- Disabling deferred writing in operation causes IBM Safer Payments to dump all data not yet saved in DDC at once. This operation will disrupt computation and can take a few minutes, depending on the amount of data that must be written.

- In case of a non-orderly shutdown, all data not saved from MDC to DDC is lost. If the respective IBM Safer Payments instance is restarted later, this missing data will not automatically be restored. If you need to restore this data, you will have to restore the entire IBM Safer Payments instance.

back to top

8.20.4 Case Investigation

The settings of this section involve the case investigation workflow build into IBM Safer Payments.

Configuration involves the following settings:

- **Manual fraud value**
  If no categories are defined for the "fraud" meta attribute, this value is assigned to the records that are manually marked as fraudulent.
  - **Remote fraud flag retries**
    Number of retries, if the remote fraud flag couldn't be set on the remote instance, for example due to FLI synchronization issues. 0 means no re-try, just send the fraud marking once.
    - **Remote flag retry each**
      Number of seconds to wait each time the remote fraud flag couldn't be set on the remote instance, for example due to FLI synchronization issues. While waiting, the FLI is not receiving further FLI message from the instance that was previously trying to set the fraud flag.
    - **Update calendar profiles and events by manual fraud marks**
      When enabled, setting manual fraud marks will update calendar profile periods or events. This can have effects on a calendar profile when it uses "fraud" attribute as "amount attribute" or in its conditions. It can effect an event when it uses "fraud" attribute in its conditions.
    - **Maximum cases shown on selection table**
      Maximum number of cases that are sent from the IBM Safer Payments server to the browser as a result of a case search or a case selection. This is to limit the load time of the page.
    - **Maximum cases shown in history**
      Maximum number of cases that are sent as case history query of the case investigation page. This is to limit the load time of the page.
    - **Archive cases after [days]**
      Automatically archives cases that have been generated and not been worked on since at least the defined number of days.
    - **Clear reporting attribute caches after [days]**
      Automatically clears the reporting attributes cache of cases that have been generated and not been worked on since at least the defined number of days. The cache is built when viewing cases. It contains attributes that have been added to the case class after the case has been created and attributes from other case classes when case selection is viewed for several case classes with different sets of attributes. Only attributes from other case classes are cleared.
    - **Case consolidation starts every [seconds]**
      Enter a time period in seconds in which case consolidation job should periodically start.
    - **Case escalation starts every [seconds]**
      Enter a time period in seconds in which case escalation job should periodically start.
    - **Case dispatching starts every [seconds]**
      Enter a time period in seconds in which case dispatching job should periodically start.
    - **Enable attachments**
      If checked, investigators can attach files to cases from their computers.
    - **Maximum case attachment size**
      If case attachments enabled, this parameter defines the maximum size of attached files in MB.
    - **Include DDC in case creation**
      If checked, data available on disk will be included when creating cases from query.
    - **Case aggregation history**
      If checked, aggregated alarms will be stored in the case.

    Case archiving details

    IBM Safer Payments creates cases from alarms. They are both created internally in RAM and on disk as a separate file for each case. All cases are stored in subdirectories of the "inv" directory, whose location is defined by a registry entry for IBM Safer Payments. Within this "inv" directory, there is an "INVxxx" and "ARCxxx" subdirectory for each mandator. "xxx" is the unique ID of the mandator. There is a readme text file within each directory detailing the originating mandator. The "INVxxx" directories contain the active cases, while the "ARCxxx" directories contain the archived cases.

    Each case file name follows the convention (example):

    ```
    investigation_case_2011-08-08_10-53-13_000000000000441700.iris
    ```

    (The timestamp value denotes the case generation and the number is the unique internal case ID of IBM Safer Payments).

    The format in which cases are stored is JSON, like all non-binary IBM Safer Payments files. Unlike the IBM Safer Payments configuration files, however, in case files, the references to any IBM Safer Payments object (such as attributes, case classes, rules etc.) are also included in clear text. This enables reading the case files also outside IBM Safer Payments. The files use UTF-8 for non-English characters. To increase readability for JSON files, use a formatter such as www.jsonlint.com.

Archiving is carried out once per day in a service thread spun off by IBM Safer Payments. This thread identifies all cases that shall be archived, removes them from its main memory caches, and moves the case file from the respective "INVxxx" directory to the "ARCxxx" directory.

Remarks

- IBM Safer Payments never erases cases from the "ARCxxx" directories. It is the responsibility of the administrator to ensure that files in these directories are moved to a safe place or deleted after they are not needed anymore.
- Once a case file is archived, IBM Safer Payments cannot display it or search for it anymore. Also its audit trail entries are not accessible anymore from within IBM Safer Payments.

back to top

8.20.5 SNMP Settings

IBM Safer Payments supports SNMP (Simple Network Management Protocol).

back to top

8.20.6 Query

The following settings configure queries in general to avoid displaying problems and excessive query computation length:

- **Records warning at**
  Some (mostly older) browsers have problems loading query tables with very large content. For this reason, the number of records may be limited for the display. If a user defines a query in which the number of records exceeds this value, IBM Safer Payments will show a warning dialog.
- **Records limit at**
  To avoid excessive query computation length this value defines queries maximum record limit. If a user defines a query in which the number of records exceeds this value the validator will show an error.
- **Group by query accounts limit**
  Some (mostly older) browsers have problems loading query tables with very large content. For this reason, the number of records may be limited for the display. IBM Safer Payments will never print more accounts than this number, even if a group by query result contains more.
- **DDC may be enabled**
  If checked, users may define queries that also use data from the disk data cache (DDC). Attributes which are only stored in ddc, may be used within queries and their conditions. This may result that a query computation require more time.
- **Result lifetime**
  Defines the time a query result is stored. After expiration of this time period, IBM Safer Payments has to recompute query results.
- **Common point result lifetime**
  Defines the time a common point query result is stored. After expiration of this time period, IBM Safer Payments has to recompute query results.
- **CSV export nil value**
  Defines which value should be printed in query result exports in case there is either no value provided or it is not accessible in MDC/DDC. There is an individual setting for data type numeric, text and timestamp. If "Original value" is chosen, the usual value for CSV exports in IBM Safer Payments is printed, which is "0" for empty numeric values, "" for empty text and timestamp and "" when a value is not available in MDC/DDC. Choosing "Empty value", no value will be printed for empty values and values not available in MDC/DDC.

Single API queries provide an external interface to IBM Safer Payments index query results. They are sent with HTTP GET, have no session, no CSRF protection and need to validate the password on every query. If they are not used, they should be disabled in IBM Safer Payments. If needed, it is recommended to restrict them to system users (see user accounts).

The following settings restrict access to single API query configuration:

- **Access to single API query**
  The access can be enabled for all users, restricted to system users or completely disabled

back to top

9. Cluster

This section covers cluster management related administration functions of IBM Safer Payments.

back to top

9.1 Cluster Settings

This page describes the cluster settings table. General information is found on the IBM Safer Payments cluster management page.

Table rows

Each instance of the IBM Safer Payments cluster is shown in one table row. For each row, the values in the columns show various status information (details below). A left click on a row opens a detailed form with all relevant settings for this instance; a right click opens a context menu with shortcuts to frequently used actions.

Table actions

While all actions on an IBM Safer Payments instance are available from the instance form that opens when you left click on the respective row, a right click opens a context menu with shortcuts for the respective IBM Safer Payments instances. Refer to operational cluster control for details.

Table columns

- **Instance Id**
  IBM Safer Payments instances are numbered from 1. The instance that you are currently using to access this page is marked with an asterisk (*).
- **Status**

  This value can be of:

  - 🟥 **Unreachable**
    Instance is not running or cannot be reached by the current IBM Safer Payments instance.
  - 🟥 **[Unreachable|Invalidated] (detached)**
    Instance has been detached. No outgoing FLI queues have been created for this instance. This instance is out of sync and needs to be restored.
  - 🟨 **Startup**
    Instance is currently starting (this may take a few minutes since during this phase, IBM Safer Payments loads its memory data cache from the disk data cache). In this status, the IBM Safer Payments instance has no open interfaces with the exception of the status and command interface (SCI) and is not responding to direct user interface requests (you will thus only see this status from another IBM Safer Payments instance). This is a transient status. If it remains for an unusual long time, check system health and event log message files. Once all startup actions are completed, the IBM Safer Payments instance will attempt synchronization of its data repository from the other IBM Safer Payments instances.
  - 🟨 **Synchronising**
    Instance tries to obtain missing transaction and configuration data from other cluster instances. You should see the progress of this effort from the FastLink status table (above this table). If the links are down, ensure that the FastLink interface (FLI) on the synchronizing instance is enabled and active.
  - 🟩 **OK**
    Instance is up and running.
  - 🟨 **Waiting for synchronisation**
    Same as startup, but with a hotstart rather than a cold start (hotstart can for instance be triggered from this table's action menu; in contrast to a cold start, the IBM Safer Payments process is not terminated).
  - 🟥 **Error**
    Instance startup failed not available due to a severe error during startup. Check event log message files for details.
  - 🟥 **Invalidated**
    A failure on this instance has caused its data to be corrupted. To protect integrity, this instance has closed its interfaces. You need to restore this instance from another one. For details, see restore process.
  - 🟨 **Lockdown**
    There are 2 ways that an instance can go into lockdown status. The first is during a restore: once a restore is started, the donor instance closes all its interfaces and waits until all its outgoing FastLink interface (FLI) buffers are drained. During this period, the donor IBM Safer Payments instance is in lockdown status. For details, see restore process.
    The second way an instance can go into lockdown status is if the FLI buffer reaches capacity and overflows. In this case the instance will lockdown to prevent further data loss on other instances. It will change again to a healthy status after restart. To recover the data loss on the other instances, it is recommended to restore all other instances with that instance selected as donor, which previously had the lockdown status. For details, see restore process.
  - 🟨 **Restoring: donor**
    This is the donor instance of a restore process and a restore is currently under way. For details, see restore process.
  - 🟨 **Restoring: recipient**
    This is the recipient instance of a restore process and a restore is currently under way. For details, see restore process.
  - 🟥 **Restore failed**
    Restore failed on this instance. Check event log message files for reasons. For details, see restore process.
  - 🟨 **Starting services**
    This is (typically) a short transient status during startup where IBM Safer Payments spins of its various service threads. If this status remains for an unusual long time, check system health and event log message files.
  - 🟨 **Undetermined**
    Startup did not conclude with a "real" status. Check system health and event log message files.
  - 🟨 **Waiting for key**
    The instance cannot start since encryption keys are not entered and activated, and cannot be obtained from other IBM Safer Payments instances. Enter and activate keys to continue the startup process.
  - 🟨 **Offline**
    All interfaces except Encrypted Communication Interface (ECI) and Status Control Interface (SCI) are inactive and all pending data has been written to disk. This state is suitable to create file backups.

- **Name**

  Name defined for this IBM Safer Payments instance.
- **Comment**

  Comment defined for this IBM Safer Payments instance.
- **Message command interface (MCI)**

  Status of MCI (Message and Command Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.
- **Application programming interface (API)**

  Status of API (Application Programming Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) and number of active user sessions are provided in brackets.
- **Batch data interface (BDI)**

  Status of BDI (Batch Data Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.
- **FastLink interface (FLI)**

  Status of FLI (FastLink Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.
- **Encrypted communication interface (ECI)**

  Status of ECI (Encrypted Communication Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail.

• **Alert message interface (AMI)**

Status of AMI (Alert message interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail.

back to top

9.1.1 FastLink Status

The pivot table of this section shows the status of the FLI (FastLink Interface) connections between the IBM Safer Payments instances of a cluster.

Each row represents the sending instance, while each column represents a receiving instance. Since no instance connects to itself, the diagonal fields are empty. The other fields show the status of the respective connections. A colour coding scheme enables a quick overview:

- 🟥 This represents a link which is currently down. Detailed information is provided on how much data is currently held in the FLI buffer, and if there is any data for which the target has not yet acknowledged receipt.
- 🟨 This represents a link which is still transmitting data, yet the receiving instance is slower in receiving (and processing) the replicated messages than they are build up. There is also detailed information about how many messages are stored and how many are outstanding and not yet acknowledged.
- 🟩 Like before, but the number of stored/outstanding messages is below the threshold defined on the system configuration page at which IBM Safer Payments considers an FLI "synchronized".
- No colour icon represents a link that IBM Safer Payments cannot determine the status for. E.g. the respective IBM Safer Payments instance may not be reachable.

The detailed information in each field is:

- Not acknowledged: number of messages that the sending IBM Safer Payments instance has already transmitted, yet not received acknowledgment of their receipt.
- Buffered: number of unsent messages in the outgoing buffer of the sending IBM Safer Payments instance.
- Total: number of all stored messages in the outgoing buffer of the sending IBM Safer Payments instance (buffered plus not acknowledged).
- Used: percentage of memory of outgoing buffer of the sending IBM Safer Payments instance used.

Notice that all information displayed in this table is transmitted via SCI and thus current. To update the display, you can select "cluster" again from the navigation menu left. You may also set an auto-refresh interval on the system configuration page. There are no controls in this section, all data is display only.

back to top

9.1.2 Cluster Instance

The IBM Safer Payments instance form both lets you configure and control an instance in a cluster. Control is provided by the toolbar actions that are described on the operational cluster control help page.

Form settings

The form lets you define a number of settings for this instance of an IBM Safer Payments cluster. Notice that most settings do not become effective immediately. The settings are:

- **Instance Id**
  Unique identification number for each IBM Safer Payments instance in a cluster. Typically starts with 1 and is incremented for each instance.
- **Name**
  Name of this IBM Safer Payments instance (choose it to differentiate the instance's location).
- **Comment**
  Description of this IBM Safer Payments instance (e.g. its physical and/or logical location, its function, etc.).
- **MCI (Message and Command Interface) settings**
  Typically the MCI is active on all IBM Safer Payments instances in a cluster to allow for the connected system to use either of the instances (redundancy). If you intend to take down one IBM Safer Payments instance of a cluster, you might want to disable the MCI interface before so that the connecting systems will stop sending this instance transaction request messages. (If you just shut down an instance without closing the MCI before, it will automatically be closen on shutdown.) Because the MCI is an IP based interface, you must define IP address and port for it (the definition of an IP address for each IBM Safer Payments interface supports server hardware with multiple network interfaces). The MCI also supports connection filtering that is enabled by unchecking the "all connections" box. If unchecked, an entry field opens that lets you enter a (comma separated) list of IP addresses for which the MCI accepts connections.
- **API (Application Programming Interface) and user interface settings settings**
  The API is fully active only on one IBM Safer Payments instance in a cluster. However, even on a disabled API, each IBM Safer Payments instance still serves basic administrational capabilities so that in case the currently API enabled instance of an IBM Safer Payments cluster becomes unavailable, the administrator can use each other instance of the cluster to perform administration tasks and even to fully enable the API interface of this instance. If you intend to take down the IBM Safer Payments instance of a cluster that has the API enabled for user access, you might want to enable the API on another instance so that the API on this instance automatically gets disabled. Because the API is an IP based interface, you must define IP address and port for it (the definition of an IP address for each IBM Safer Payments interface supports server hardware with multiple network interfaces). The API also supports connection filtering that is enabled by unchecking the "all connections" box. If unchecked, an entry field opens that lets you enter a (comma separated) list of IP addresses for which the API accepts connections. Because the API uses the HTTP protocol, the port is typically "80".
- **BDI (Batch Data Interface) settings**
  The BDI is also only fully active on one IBM Safer Payments instance in a cluster. In the case that the IBM Safer Payments instance where the BDI is active is to be taken down, similar to the API, the BDI of another IBM Safer Payments instance should be enabled first, so that the batch data processing function is carried out by the other instance.
- **FLI (FastLink Interface) settings**
  The incoming FLI is where an IBM Safer Payments instance receives the replication messages of all other IBM Safer Payments instances of the cluster. Disabling the FLI of an instance causes the replication messages being temporarily buffered in the outgoing FLI of the other IBM Safer Payments instances. Outgoing FLI are not explicitly defined as the cluster definition is available to all IBM Safer Payments instances and this tells the instance which other instances to connect to with their outgoing FLI connections.
- **SCI (Status and Control Interface) settings**
  The SCI is always active on all IBM Safer Payments instances in a cluster. This is because the SCI is the central lifeline between the IBM Safer Payments instance to exchange health and status information. As an IP interface, you may define IP address and port here. The SCI will only accept connections from the other (defined) IBM Safer Payments instances, thus no connection filtering settings are necessary.
- **ECI (Encrypted Communication Interface) settings**
  The ECI is used between IBM Safer Payments instances of a cluster to exchange AES encryption keys. It is not needed for an unencrypted IBM Safer Payments installation. If deactivated, no exchange of encrypted keys is possible. As an IP interface, you may define IP address and port here. The SCI will only accept connections from the other (defined) IBM Safer Payments instances, thus no connection filtering settings are necessary.
- **AMI (Alert Message Interface) settings**
  The AMI is an outgoing interface for alert messages. It can be active on all instances. If deactivated, a generated alert message (e.g. SAI or email) will be routed to another instance with active AMI. If there is no active AMI, the alert message will be discarded.

IP Addresses and Ports

All interfaces (API, ECI, FLI, MCI and SCI) of all cluster instances must use a unique combination of IP address and port. This is checked server-side to avoid invalid configurations. Host names are resolved to IP addresses but we recommend to directly use IP addresses to make the cluster operation independent of the DNS. When choosing ports for the interfaces it is recommended to not choose a port in the ephemeral port range. You can check the configured range on your system by running `sysctl -A | grep ip_local_port_range`, and can update the range if necessary. It is especially important to not use this range in test environments if you have multiple Safer Payments instances on a single machine. Both IPv4 and IPv6 addresses are valid but ECI, FLI and SCI must each use the same protocol version on all instances to function properly. The loopback addresses '127.0.0.1' and '::1' are only considered equal to themselves (not to each other or any other local IP address). Link-local IPv6 addresses (addresses starting with fe80::) are not supported.

Accepting Connections From Specified Systems

The API and MCI can be set up to only accept connections from specified hosts or addresses. Both IPv4 and IPv6 addresses as well as host names can be used. When the external system has several IP addresses, we recommend to either enter all of them or use the host name if possible.

Change Certificates

- Copy new certificate to ./key folder - do not overwrite old certificates.
- Open cluster instance settings of currently active IBM Safer Payments instance
- Change paths of certificate
- Save instance

back to top

9.2 Cluster Management

IBM Safer Payments is "cluster-ready" out of the box. This help page introduces you to the issues involved with cluster configuration and operation.

Availability

For most applications, a single IBM Safer Payments instance is sufficient because IBM Safer Payments can run 24/7 without any maintenance or batch window (batch operations and maintenance are spun off as separate services by IBM Safer Payments and are executed fully in parallel to operations). While the systems connected to IBM Safer Payments with a real-time interface (for instance, authorisation systems) need a response from IBM Safer Payments within milliseconds to include IBM Safer Payments advice on intercepting with a high-risk authorisation request, they continue after a timeout period with their operation without the IBM Safer Payments advice if IBM Safer Payments does not respond. Thus, non-availability of IBM Safer Payments never halts the authorisation process.

Availability business case

However, while IBM Safer Payments is not available, fraud otherwise detected by IBM Safer Payments would go unprevented while IBM Safer Payments is not available. Availability of the IBM Safer Payments system thus becomes a cost comparison between the cost of increasing availability and the losses associated with fraud that occurs during the non-availability period. Assuming your platform with IBM Safer Payments would provide 99% availability, which translates to 87.6 hours of downtime per year, and IBM Safer Payments saving 50% of your losses of – without IBM Safer Payments – $10 Million per year, the fraud loss increase resulting from the 1% downtime is $50,000. With 99.9% availability, this would equal to losses of $5,000 and with 99.999% availability it would be $50.

Because the diminishing returns with higher availability and the over proportionally increasing cost of providing systems with such a low downtime, a decision on which level of availability is best for your enterprise can only be made using the numbers for your enterprise.

It is also far from trivial to compute the estimated downtime for a given setup. The process involves experience, educated guesses and often comes out as somewhat of a "black art".

Numbers game

The way IBM Safer Payments provides higher levels of availability is by setting up multiple IBM Safer Payments systems on multiple computer servers, and connecting them to a cluster. Because each IBM Safer Payments instance in an IBM Safer Payments cluster must be sized to process the full system load, all IBM Safer Payments functions are available as long as one IBM Safer Payments instance is available. Assuming three IBM Safer Payments instances with an availability of each 99% and the cause of outages random, the probability that all three IBM Safer Payments instances are down at the same time is 0.0001% ($(100\% - (100\% - 99\%)^3) = 99.9999\%$).

All availability calculations in this section are only to illustrate the basic considerations. In real world cases, calculations of availability are far more complex. However, the general rule that more IBM Safer Payments instances provide a higher level of availability stands.

There also might be other reasons for using multiple IBM Safer Payments instances, such as that you may have two active datacenters each running an authorisation system, and a third one as disaster recovery. In such a situation it would be imperative to have an IBM Safer Payments instance in each datacenter; regardless of availability considerations.

Performance gains

In most other applications, clustering is a technique to increase the throughput performance of a system. With IBM Safer Payments applications, this is typically not the case. Because IBM Safer Payments makes its decisions based on past transaction data, each IBM Safer Payments instance must store and compute all transactions. If you for instance have two authorisation systems in two datacenters each with a "local" IBM Safer Payments instance, and you route transactions "round robin" to each of these authorisation systems, the two IBM Safer Payments instances need to update each other on the transactions they missed out on. Because this "background update" process requires about the same computational resource in IBM Safer Payments as processing the transaction, adding IBM Safer Payments instances in a cluster does not increase total throughput performance.

Because transaction processing throughput of a single IBM Safer Payments instance on standard hardware already suffices for even ultra large scale applications, this non-scaling-up characteristic of an IBM Safer Payments cluster is not typically a drawback.

Operations and control

Using multiple IBM Safer Payments instances in a cluster is completely supported by IBM Safer Payments. Each IBM Safer Payments instance contains the administrational functionality to control all instances in a cluster. The administrational processes associated with the operation of an IBM Safer Payments cluster are described below. First the installation of an IBM Safer Payments cluster is described.

IBM Safer Payments instances

While IBM Safer Payments is capable of running in a cluster with any number of IBM Safer Payments instances, the practical numbers of IBM Safer Payments instances in a cluster are 1, 2, or 3. One is not really a cluster, but in many applications, a single IBM Safer Payments instance suffices. Also notice that one IBM Safer Payments instance can support any number of authorisation systems. If you are using one IBM Safer Payments instance, you will have downtime in the following cases:

1. Computer server hardware malfunction
2. Operating system malfunction
3. IBM Safer Payments update
4. IBM Safer Payments software malfunction

On the subject of updates, notice that with "normal" updates, all you need to do is to stop IBM Safer Payments, install the new release over the existing one, and restart it. The "new" IBM Safer Payments release will come up exactly where the old one has left off. Only major releases may imply additional administrator actions. In this case, we would inform you about this well in advance, and assist you, if required.

Using two IBM Safer Payments instances provides you with full redundancy in standard operations. While one IBM Safer Payments instance is unavailable, the other IBM Safer Payments instance takes over full load. Once the second IBM Safer Payments instance becomes responsive again, the first IBM Safer Payments instance updates ("synchronises") the second IBM Safer Payments instance on what it missed out on. The operational details of this are explained below. Using two IBM Safer Payments instances also allows you to have scheduled downtime, for instance to replace computer hardware, update the operating system or update IBM Safer Payments without any service interruption.

However, during scheduled downtime, IBM Safer Payments does not operate redundantly. If you take down one IBM Safer Payments instance, and the other one fails during this period, IBM Safer Payments service is interrupted. To have redundant operations even in this case, three IBM Safer Payments instances are needed. Using three instance has the additional advantage that after a complete loss of an individual instance (e.g. because of hardware failure) it can be restored from another instance without service interruption (see below). During restoration, the third instance ensures full availability of IBM Safer Payments.

Whilst you may use four or more instances, there typically is no practical advantage to be gained from this.

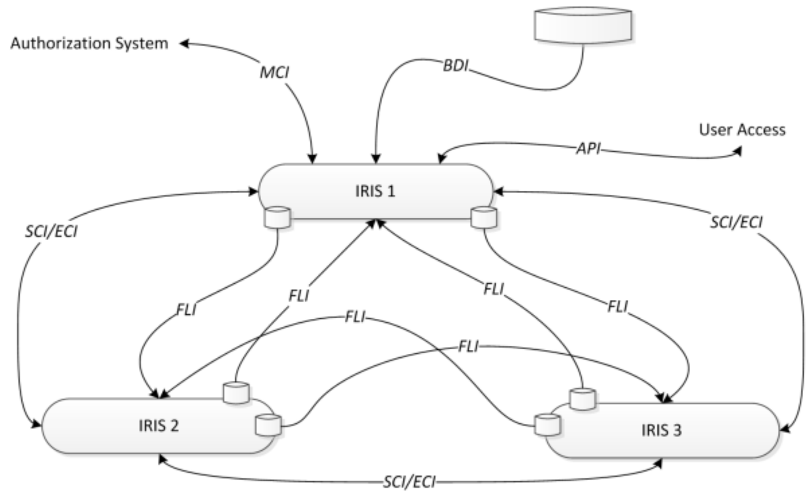
Maintenance

To make cluster management as easy and integrated as possible, all maintenance functions are provided from the same IBM Safer Payments pages as all other user access. Therefore even IBM Safer Payments instances whose API is not enabled, provide access to the cluster administration page. Hence each IBM Safer Payments instance in a cluster can be used to manage the entire cluster.

If the one IBM Safer Payments instance with the API enabled should fail, this feature ensures that as long as there is still one operating IBM Safer Payments instance in the cluster, the administrator can use this instance to perform all maintenance tasks.

Normal operations

This section exemplifies a configuration of an IBM Safer Payments cluster with 3 instances. The sketch below shows the 3 instances with their external and internal interfaces. The external interfaces are the MCI (Message and Command Interface) for online transaction request/responses, the BDI (Batch Data Interface) for transaction message file delivery, and the API (Application Programming Interface) for user access. Both BDI and API can only be active on one of the IBM Safer Payments instances at the same time, though they can each be set active on different instances. The example assumes that both interfaces are activeon instance 1 only.



The MCI can be enabled on each instance and each IBM Safer Payments instance can assume the full transaction load as delivered by the authorisation system. This is the normal operating condition. The authorisation system must have functionality to turn to the next IBM Safer Payments instance in the cluster once it has determined that the IBM Safer Payments instance it currently uses is not responsive or down. Not-responsiveness is typically detected by a watchdog timer in the authorisation system that kicks in once a transaction request message has not been responded to within an allotted timeframe. If the authorisation system experiences that the IBM Safer Payments instance it connects to drops the connection, it can assume that the IBM Safer Payments instance or the network route to it went down. In this case, the authorisation system should immediately turn to the next instance.

Notice that each IBM Safer Payments instance has three internal interfaces, the SCI (Status and Control Interface) for status and control commands, the ECI (Encrypted Communication Interface) to exchange encryption keys and login credentials, and the FLI (FastLink Interface) for transaction and configuration data. Because the FLI must tolerate connection or instance outages, each outgoing FLI interface is FIFO disk buffered. The size of the FLI buffer files are fixed and defined on the system configuration page. The best size is determined by the transaction message frequency in relation to the outage time the buffers shall cover.

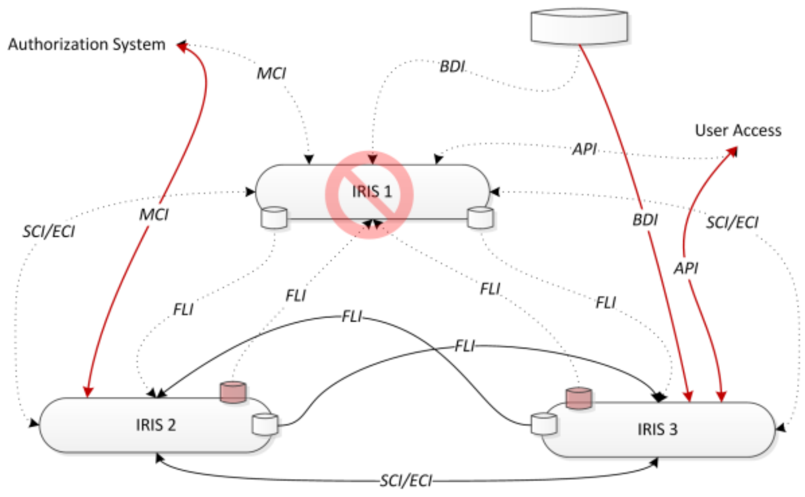
The SCI and ECI are not buffered since a FIFO buffer would not guarantee that the status information provided is current. For the same reason, the replication of the cluster configuration (the contents of the "cluster.iris" file) is handled via the SCI.

Instance shutdown

Certain operational conditions require the scheduled shutdown of an IBM Safer Payments instance. This is for instance the case if the server hardware or software of one IBM Safer Payments instance is updated. Shutdown and restart of this IBM Safer Payments instance is straightforward and involves the following steps.

Assuming IBM Safer Payments 1 is to be shut down, the administrator disables the MCI, BDI, API and incoming FLI interfaces. The BDI interface will, upon being disabled, complete the computation of the currently processed record (for each job running), and then interrupt all jobs. They will later start at the next record position when the BDI is enabled again on any IBM Safer Payments instance connected. The other interfaces, upon being disabled, will refuse any connection opening requests, will complete the computation of the currently processed transaction message, and then shut down the connections that were open.

In this example, it is further assumed that BDI and API were activated on IBM Safer Payments 3, while the authorisation system chooses to route its transaction message requests to IBM Safer Payments 2 when it finds the MCI interface on IBM Safer Payments 1 disabled. When the FLI incoming interface of IBM Safer Payments 1 disables, the FLI outgoing interfaces of IBM Safer Payments instances 2 and 3 receive a connection close from IBM Safer Payments 1 and start buffering all transaction message copies and all configuration data copies into their file buffers.



Once IBM Safer Payments 1 is restarted, and its incoming FLI is activated, IBM Safer Payments instances 2 and 3 automatically discover this as they are periodically testing IBM Safer Payments 1 for responsiveness. Once they find IBM Safer Payments 1 to be receiving FLI data again, they will feed all buffered transactions to the IBM Safer Payments 1 instance in sequence.

Once "synchronised", that is, IBM Safer Payments 1 being up to date with transaction and configuration data, the MCI, API, and BDI interfaces of IBM Safer Payments 1 can be activated, if IBM Safer Payments 1 should now re-assume these tasks at this time.

Instance failure (hot start)

While the example before assumed a scheduled shutdown of an IBM Safer Payments instance, the recovery process of a non-scheduled shutdown (hardware, or software failure) of an IBM Safer Payments instance is actually pretty much the same.

The main difference is that in case that for instance IBM Safer Payments 1 experiences a non-scheduled shutdown, the BDI and API interfaces have not been moved to other IBM Safer Payments instances. Therefore in this case, BDI and API instantly become unavailable. For BDI, this condition is simple to recover from as the BDI resumes operation always from the point it last stopped, whether this stop was intentional (administrator command) or unintentional (instance failure). For the API, this actually results in the user sessions being canceled. Data a user had not saved at this moment is lost. Because the amount of data a user can enter before he must save is kept to a practical minimum in IBM Safer Payments, the loss of data should be tolerable as the user typically remembers rather well what he just had entered.

The process for recovery is simple: the administrator logs onto either of the remaining operational instances, moves BDI and API to an operational IBM Safer Payments instance, and full operation resumes.

If a second instance becomes non-available during downtime, the recovery process is the same, only that once the last operational IBM Safer Payments instance fails, IBM Safer Payments cluster operation is down.

As soon as any IBM Safer Payments instance comes back operational, the remaining instances automatically synchronise this IBM Safer Payments instance with the contents of their FLI buffers, and once synchronised, the IBM Safer Payments instance becomes a fully functional member of the cluster again.

Instance failure (cold start)

In the cases discussed before, whether scheduled or non-scheduled shutdown of an IBM Safer Payments instance occurs, it is assumed that the data on disk of these instances survives. This would normally be the case. In case of a scheduled shutdown, where typically part of the infrastructure is replaced or the operating system or IBM Safer Payments itself is updated, disk data remains intact. In case of a non-scheduled shutdown, data is rarely lost because IBM Safer Payments instances are typically installed on fault-tolerant RAID disk subsystems.

There are, however, rare possible situations in which disk data on an IBM Safer Payments instance can be lost. In this case, the recovery processes described before cannot be used because the data stored in the outgoing FLI buffers would not be sufficient to synchronise the IBM Safer Payments instance. In this case, a more complex ("cold start") recovery process must be employed.

Also, there is the situation of the FLI buffer exhaustion of the other IBM Safer Payments instances. This can happen if during the outage of an IBM Safer Payments instance the amount of data stored for synchronisation in the FLI buffers exceeds the allocated buffer size. In this case, the FLI buffers cannot restore the non-active IBM Safer Payments instance anymore when it would start up later, so the FLI buffers are dropped, and the non-active IBM Safer Payments instance must be "cold started". Notice that there are buffer management functions within the FLI to prevent this situation from occurring.

In the 3 instance configuration discussed in this example, if one IBM Safer Payments instance disk data is lost, or the FLI buffer of another instance feeding this IBM Safer Payments instance are dropped, this IBM Safer Payments instance must be "restored". This process and related other operational processes are described in detail on the online help cluster instance page.

back to top

9.2.1 Cluster Settings

This page describes the cluster settings table. General information is found on the IBM Safer Payments cluster management page.

Table rows

Each instance of the IBM Safer Payments cluster is shown in one table row. For each row, the values in the columns show various status information (details below). A left click on a row opens a detailed form with all relevant settings for this instance; a right click opens a context menu with shortcuts to frequently used actions.

Table actions

While all actions on an IBM Safer Payments instance are available from the instance form that opens when you left click on the respective row, a right click opens a context menu with shortcuts for the respective IBM Safer Payments instances. Refer to operational cluster control for details.

Table columns

- **Instance Id**
  IBM Safer Payments instances are numbered from 1. The instance that you are currently using to access this page is marked with an asterisk (*).
- **Status**

  This value can be of:

  - 🟥 **Unreachable**
    Instance is not running or cannot be reached by the current IBM Safer Payments instance.
  - 🟥 **[Unreachable|Invalidated] (detached)**
    Instance has been detached. No outgoing FLI queues have been created for this instance. This instance is out of sync and needs to be restored.
  - 🟨 **Startup**
    Instance is currently starting (this may take a few minutes since during this phase, IBM Safer Payments loads its memory data cache from the disk data cache). In this status, the IBM Safer Payments instance has no open interfaces with the exception of the status and command interface (SCI) and is not responding to direct user interface requests (you will thus only see this status from another IBM Safer Payments instance). This is a transient status. If it remains for an unusual long time, check system health and event log message files. Once all startup actions are completed, the IBM Safer Payments instance will attempt synchronization of its data repository from the other IBM Safer Payments instances.
  - 🟨 **Synchronising**
    Instance tries to obtain missing transaction and configuration data from other cluster instances. You should see the progress of this effort from the FastLink status table (above this table). If the links are down, ensure that the FastLink interface (FLI) on the synchronizing instance is enabled and active.
  - 🟩 **OK**
    Instance is up and running.
  - 🟨 **Waiting for synchronisation**
    Same as startup, but with a hotstart rather than a cold start (hotstart can for instance be triggered from this table's action menu; in contrast to a cold start, the IBM Safer Payments process is not terminated).
  - 🟥 **Error**
    Instance startup failed not available due to a severe error during startup. Check event log message files for details.
  - 🟥 **Invalidated**
    A failure on this instance has caused its data to be corrupted. To protect integrity, this instance has closed its interfaces. You need to restore this instance from another one. For details, see restore process.
  - 🟨 **Lockdown**
    There are 2 ways that an instance can go into lockdown status. The first is during a restore: once a restore is started, the donor instance closes all its interfaces and waits until all its outgoing FastLink interface (FLI) buffers are drained. During this period, the donor IBM Safer Payments instance is in lockdown status. For details, see restore process.
    The second way an instance can go into lockdown status is if the FLI buffer reaches capacity and overflows. In this case the instance will lockdown to prevent further data loss on other instances. It will change again to a healthy status after restart. To recover the data loss on the other instances, it is recommended to restore all other instances with that instance selected as donor, which previously had the lockdown status. For details, see restore process.
  - 🟨 **Restoring: donor**
    This is the donor instance of a restore process and a restore is currently under way. For details, see restore process.
  - 🟨 **Restoring: recipient**

This is the recipient instance of a restore process and a restore is currently under way. For details, see restore process.

  - 🟥 **Restore failed**
    Restore failed on this instance. Check event log message files for reasons. For details, see restore process.
  - 🟨 **Starting services**
    This is (typically) a short transient status during startup where IBM Safer Payments spins of its various service threads. If this status remains for an unusual long time, check system health and event log message files.
  - 🟨 **Undetermined**
    Startup did not conclude with a "real" status. Check system health and event log message files.
  - 🟨 **Waiting for key**
    The instance cannot start since encryption keys are not entered and activated, and cannot be obtained from other IBM Safer Payments instances. Enter and activate keys to continue the startup process.
  - 🟨 **Offline**
    All interfaces except Encrypted Communication Interface (ECI) and Status Control Interface (SCI) are inactive and all pending data has been written to disk. This state is suitable to create file backups.

- **Name**

  Name defined for this IBM Safer Payments instance.

- **Comment**

  Comment defined for this IBM Safer Payments instance.

- **Message command interface (MCI)**

  Status of MCI (Message and Command Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.

- **Application programming interface (API)**

  Status of API (Application Programming Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) and number of active user sessions are provided in brackets.

- **Batch data interface (BDI)**

  Status of BDI (Batch Data Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.

- **FastLink interface (FLI)**

  Status of FLI (FastLink Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail. Momentary load (10 second average) is provided in brackets.

- **Encrypted communication interface (ECI)**

  Status of ECI (Encrypted Communication Interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail.

- **Alert message interface (AMI)**

Status of AMI (Alert message interface) on this IBM Safer Payments instance. 🟩 if the status is as set, 🟥 if not, and no icon if interface is disabled and not active. A text next to the icon explains the status in detail.

back to top

9.2.2 Operational Cluster Control

Operating a running IBM Safer Payments cluster involves a number of activities that are each described on a separate help page:

- Add IBM Safer Payments instance
  Operational processes to add another IBM Safer Payments instance to a running cluster.
- Shutdown
  Shutdown process for a single IBM Safer Payments instance or an entire IBM Safer Payments cluster.
- Detach
  Detaching an instance from the cluster disables replication to this instance. Use it when you want to take down an instance for a longer period of time.
- Attach
  Attaching an instance forces to add an instance to the replication mechanism of the cluster, without synchronising it.
- ✕ Delete
  Permanently removes an IBM Safer Payments instance from the cluster.
- Restore
  Recreates a new or existing IBM Safer Payments instance from another IBM Safer Payments instance during operations.

Notice that setting up an IBM Safer Payments cluster, and starting it for the first time is described in the IBM Safer Payments Installation

Manual that is available from our IBM Safer Payments support site from where you can also download IBM Safer Payments releases and find related support information.

These functions are available from the cluster administration page instance form.

While an IBM Safer Payments cluster can restore its instances in operations from itself, there is also traditional backup within an IBM Safer Payments cluster.

Notice that you may also perform operational control remotely.

The golive process in a cluster is described here.

back to top

9.2.3 Add Instance

You may add an IBM Safer Payments instance to a cluster at any time during full operations. The process involves a number of steps:

1. Click [Add instance] from the cluster settings table toolbar. This opens a form to configure the new IBM Safer Payments instance. Enter the relevant information. Only activate the FastLink interface. Save your settings.
2. Install the new IBM Safer Payments instance on its platform.
3. Manually copy the file "cfg/cluster.iris" from one of the existing IBM Safer Payments instances to the new instance, overwriting the file that was part of the "empty" installation.
4. Manually copy the "key" folder if certificates or encryption keys were added. These files will not be transmitted by network.
5. Enable the ECI interface.
6. Open the file "cfg/iris.iris" with a text editor and change its contents to:
   ```
   {"iris":{"status":"New"}}
   ```
7. Start the new IBM Safer Payments instance and observe on the cluster page of another IBM Safer Payments instance its startup. Once started up, the status of the new IBM Safer Payments instance will be shown as "invalidated".
8. Perform a restore operation on the new instance to synchronize it with the other IBM Safer Payments instances.
9. Once the restore operation is complete, the new IBM Safer Payments instance has become a full member of the cluster and can now be configured to perform any of the cluster's services.

back to top

9.2.4 Shutdown

There are multiple reasons for shutting down one IBM Safer Payments instance of a cluster:

- Software or hardware maintenance of the hosting server.
- Update of an IBM Safer Payments instance.

Whatever the reason, if you follow the process steps detailed below, you will be able to restart the IBM Safer Payments instance later and not have lost any data.

Isolation

Before the IBM Safer Payments instance can be shut down, you must transfer its functions to the other IBM Safer Payments instances. How this is done, depends on the interface:

- BDI (Batch Data Interface)
  By disabling the BDI on the instance you want to shut down, all data from file loading processes are halted. To resume them from another IBM Safer Payments instance, enable the BDI on this instance. If you have not disabled the BDI on the first IBM Safer Payments instance already, enabling the BDI on the second instance will automatically disable it on the first one (to ensure that at each time only one IBM Safer Payments instance performs batch data loading).
- API (Application Programming Interface)
  By disabling the API on the instance you want to shut down, all user sessions on this API are terminated. This excludes users with cluster administration privileges for cluster control functions, so while you are performing a shutdown process, this will not affect your session. To allow user access from another IBM Safer Payments instance, enable the API on this instance. If you had not disabled the API on the first IBM Safer Payments instance already, enabling the API on the second instance will automatically disable it on the first one (to ensure that at each time users access IBM Safer Payments only through one IBM Safer Payments instance).
- FLI (FastLink Interface)
  By disabling the FLI on the instance you want to shut down, all replication data from the other IBM Safer Payments instances are now buffered within their FLI outgoing queue buffers. This ensures that when the IBM Safer Payments instance that you want to shut down starts up again, and FLI is enabled again, the buffers deliver all the production and configuration data to this IBM Safer Payments instance it had missed when it was not running.
- MCI (Message and Command Interface)
  By disabling the MCI on the instance you want to shut down, all service consumers re-route their transaction message requests to another IBM Safer Payments instance in the cluster. Notice that if you shut down an IBM Safer Payments instance without having disabled the MCI before, the MCI will be disabled automatically during shutdown. It is, however, still a good policy to first manually disable the MCI before shutting down an instance. For example, if the MCI is automatically disabled during shutdown, there is a

grace period IBM Safer Payments waits for all transactions to be terminated before it closes all MCI connections. This grace period is typically defined long enough that all computation is completed, yet if for whatever reasons, it is not, you may interrupt computation of a transaction before it is completed.

Once you have disabled all these interfaces to IBM Safer Payments, no new production or configuration data is generated in this instance. However, since data could have been built in the FLI outgoing queues of *this* IBM Safer Payments instance, you must check the FastLink status section on the cluster administration page to ensure that all outgoing queues of this instance have emptied.

Shutdown sequence

Now you may initiate shutdown by clicking on the respective ● [Shutdown] tool icon on the cluster instance form of this instance, or from the context menu of the respective row of the instance in the cluster settings table. After clicking on the respective button, you will see a shutdown dialog. You can select if you want to shutdown this instance immediately or at a particular time. If you want to shutdown this instance earlier, you may select an earlier shutdown. Please notice that a started shutdown cannot be interrupted.

The actual shutdown sequence empties all internal buffers of IBM Safer Payments and terminates its service threads before it ends the main IBM Safer Payments instance process. This entire sequence typically takes a few seconds.

The cluster settings table should show the status "unreachable" for this instance with a 🟥 red icon once the sequence is completed.

You should now inspect the system event log to verify that there were no errors during shutdown.

Remarks

- You may also initiate a shutdown on the IBM Safer Payments instance you are currently connected to. In particular, if you intend to shut down an entire IBM Safer Payments cluster, the last IBM Safer Payments instance you shut down is the one you are working on. Since in this case, it is impossible to view the system event log from within IBM Safer Payments, you should inspect the latest system event log instead on file level.
- To re-start an IBM Safer Payments instance after a shutdown (and potentially maintenance operation on the instance or its server), you cannot use any function from within IBM Safer Payments, you will have to turn to the method used in your IBM Safer Payments setup to start the instance. Notice that you have a certain time window for restarting, since the FLI outgoing queue buffers of the other IBM Safer Payments instances only have a finite capacity. If you have exceeded this capacity, and you re-start the IBM Safer Payments instance, it will detect that it cannot be syhcnronized to the current state of the other instances via FLI and require you to restore this instance.

back to top

9.2.5 Detach Instance

If you detach an IBM Safer Payments Instance, it will effectively be taken out of the IBM Safer Payments cluster replication mechanism. The outgoing FLI queues of all other instances to this instance will be closed and their contents will be discarded.

This ensures that if the detached IBM Safer Payments instance is not coming back up for an extended time, no FLI outgoing queue will overfill. When this IBM Safer Payments instance comes back, it need to be "restored" to make good for the missed replication.

The attach of an instance forces to reopen the outgoing queues without recovering the discarded contents. An attached instance will not be in sync any more.

Notice that the delete IBM Safer Payments instance operation also executes a detach.

back to top

9.2.6 Delete Instance

Deletion of an IBM Safer Payments instance first invokes detaching it from the cluster, and then to delete it from the cluster configuration.

Remarks

- You may delete an IBM Safer Payments instance of a cluster in full operations.
- The deleted IBM Safer Payments instance is only deleted from the cluster, not physically from its hardware platform. If you want to remove the IBM Safer Payments instance from its hardware platform, you need to uninstall it after you deleted it from the cluster.

back to top

9.2.7 Restore

Restores this instance from a donor. This function is needed in a number of cases:

- A new instance is added to the cluster.
- An IBM Safer Payments instance suffered from a major fault involving data loss (golive fault etc.).
- A FastLink queue to an instance broke (i.e. due to a buffer overfill situation).
- The disk subsystem of the host computer of an IBM Safer Payments instance suffered from data loss.

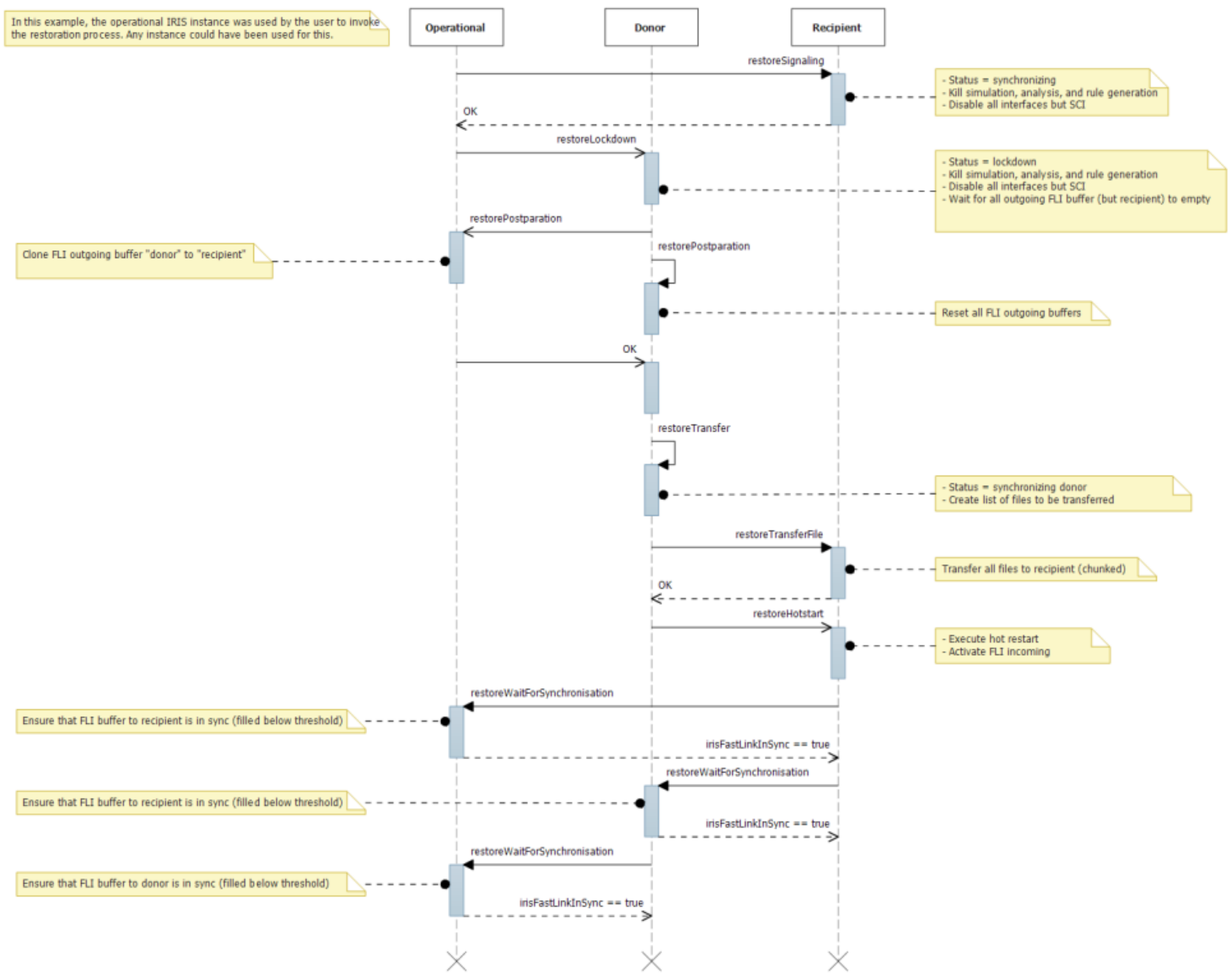
In these situations, no data on the instance can be used anymore and the instance must be restored. Restoration is a data transplantation process in which the IBM Safer Payments instance to be restored (aka "recipient") obtains all data from another IBM Safer Payments instance (aka "donor"). Notice that full IBM Safer Payments operations resume if you have 3 or more instances in a cluster because when donor and recipient are both unavailable during the restore process, there is still one more IBM Safer Payments instance to process transaction messages, batch data, and user requests (aka "operational").

Restore process

The restore process consists of a number of steps:

1. Invocation: open the edit form for the recipient instance. Click the restore button. Confirm your choice. Select the donor instance. The remained of the process is now performed automatically by the cluster. Do not interrupt until complete.
2. Signaling: within the IBM Safer Payments cluster, the instances are now informed about their role in the restoration process (donor, recipient, or operational). Within the instances table, the status of the recipient is now signaled as "Synchronisation: Recipient".
3. Lockdown: the donor closes all its incoming interfaces (message command interface, application programming interface, batch data interface, FastLink interface) and waits until its outgoing FastLink buffers are drained. The recipient status shows "Lockdown" until this part of the process is completed.
4. Postparation: all operational instances discard their FastLink outgoing buffer to the recipient and clone the FastLink outgoing buffer to the donor for the recipient. This buffer is continuously filling up during the restoration process since the donor has disabled its incoming FastLink interface for the duration. The operational instances now fill both the buffers (donor and recipient) with all data since the donor closed its interfaces. This effectively ensures that donor and recipient are both provided with the (now static) data of the donor and the (accumulating) contents of the Fast Link buffers which ensures them to be up to date and synchronized. In addition, the FastLink outgoing buffer of the donor to the recipient is discarded.
5. Transfer: all data stored on files from the donor is now copied to the recipient. This step can take considerable time, depending on the size of the data and the bandwidth of the network involved. This step is signaled by the donor status being displayed as "Synchronisation: Donor".
6. The following are carried out in parallel:
   - Hotstart: the recipient now restarts with the donor data. This step is signaled by the recipient status being displayed as "Startup".
   - Incoming FastLink interface for recipient set active so that it starts being updated by the operational instances. The progress of this step can be followed by the FastLink status display.
7. Synchronisation: incoming FastLink interfaces of donor instance activated, causing donor to be delivered with all data that was held in the FastLink buffers since lockdown of the donor. The progress of this step can be followed by the FastLink status display.
8. Finish: both donor and recipient monitor the FastLink buffers that feed them. Once they all are considered "in-sync", the respective message command interfaces are activated and the instance becomes fully operational again.

   The following UML sequence chart exemplifies the message flow between the instances during a restore procedure in detail:


In this example, the operational IRIS instance was used by the user to invoke the restoration process. Any instance could have been used for this.
Operational
Donor
Recipient
restoreSignaling
- Status = synchronizing
- Kill simulation, analysis, and rule generation
- Disable all interfaces but SCI
OK
restoreLockdown
- Status = lockdown
- Kill simulation, analysis, and rule generation
- Disable all interfaces but SCI
- Wait for all outgoing FLI buffer (but recipient) to empty
restorePostparation
Clone FLI outgoing buffer "donor" to "recipient"
restorePostparation
Reset all FLI outgoing buffers
OK
restoreTransfer
- Status = synchronizing donor
- Create list of files to be transferred
restoreTransferFile
Transfer all files to recipient (chunked)
OK
restoreHotstart
- Execute hot restart
- Activate FLI incoming
restoreWaitForSynchronisation
Ensure that FLI buffer to recipient is in sync (filled below threshold)
irisFastLinkInSync == true
restoreWaitForSynchronisation
Ensure that FLI buffer to recipient is in sync (filled below threshold)
irisFastLinkInSync == true
restoreWaitForSynchronisation
Ensure that FLI buffer to donor is in sync (filled below threshold)
irisFastLinkInSync == true

Manual restore

As an alternative to the "automatic" restore process described above, you may also restore an IBM Safer Payments instance "manually". The disadvantage of performing a manual restore is that both donor and recipient IBM Safer Payments instance must be shut down during the process. The process consists of a number of steps:

- Close interfaces on donor.
- Shut down donor and recipient.
- Delete the contents of directories "cfg", "ddc", and "inv" (and its subdirectories) on recipient.
- Copy all files of directories "cfg", "ddc", and "inv" (and its subdirectories) from donor to respective locations of recipient.
- Start both instances.

back to top

9.2.8 Backup

IBM Safer Payments cluster installation is inherently fail-resilient. Each IBM Safer Payments instance serves as a backup to all the others, and since data stored in IBM Safer Payments is typically also stored in other systems as well, there is little need to revert back to an older status of the application. Thus many IBM Safer Payments applications do not require any additional backup process.

Backup background

If this is not the case in your application, and you must implement backup processes for IBM Safer Payments, there are certain aspects to consider.

First notice that an IBM Safer Payments instance cannot be backed up during operations, because while IBM Safer Payments operates, it constantly changes file contents. In particular IBM Safer Payments requires constant and exclusive read and write access for certain files, to ensure round-the-clock real-time processing of transactions. IBM Safer Payments must be taken offline (all interfaces disabled) so that its disk data remains unchanged during any backup process. Notice that it is not necessary to shut down the IBM Safer Payments instance for backup, only all incoming interfaces have to be disabled during backup.

If your application requires periodical backups, it could be advantageous to install a separate IBM Safer Payments instance dedicated to this task. This IBM Safer Payments instance would only have FastLink interface (FLI) activated, and FLI would be suspended each time a backup from this instance is made.

An alternative to this is to use a backup software that supports "snapshot technology", able to even back up open files. Because of the typical (large) size of the IBM Safer Payments DDC files, the overhead involved with this approach could be prohibitively large.

Restoring from backup

Restoring from a backup is simple. You may also use one instance's backup to restore another or all IBM Safer Payments instances in a cluster. Make sure that you do use separate Windows registries or registry files and that they are not overwritten by the restoration process. The registry contains the information where each instance locally stores its data, and also tells the instance, which instance number within the cluster it assumes. Other than that, restoring is straightforward, simply restore all files and start the instances.

Notice that there is a restore function in IBM Safer Payments that restores one IBM Safer Payments instance from another in live operations.

back to top

9.2.9 Cluster Golive

During the actual golive of a model revision, IBM Safer Payments has to reconfigure its model and, if there were changes on the data storage configuration, the data storage objects. During this reconfiguration, IBM Safer Payments cannot compute transaction message requests. In a typical golive involving only changes of profiling and rules, this time period typically only lasts a few milliseconds. If new attributes or profiling output attributes are added to the model, this time period typically still remains in a sub-second range. In some rare cases, however, the time period can be significantly longer, for instance, if a large DDC requires encryption or decryption, or if major DDC resizes of all attributes have to be performed as part of the golive.

The golive report that is created as part of the golive process before the process actually gets started attempts to estimate this time period in advance. This is important as any golive for one mandator will exclude golives of other mandator at the same time. Typically this never results in bottleneck situations as even with hundreds of mandators, most mandator's golives will be very quick, as the mandator's users would not be privileged to perform the aforementioned major changes to the data store configuration. Also notice that the quality of the golive estimation depends on how accurate the performance factors (entered on the system configuration page) have been benchmarked for exactly the server platform used.

It is important to notice that in an IBM Safer Payments cluster setup, the non-availability of one instance for the duration of its golive does not translate to non-availability of the functioning of the entire IBM Safer Payments cluster. Assuming a three-instance cluster and a golive executed on instance 1, first instance 1 will perform golive. If this would cause transaction messages to be delayed past the timeout, the service consumer would switch sending transactions to the instances 2 or 3. Golive on instances 2 and 3 starts on both instances only after golive on instance 1 is completed and instance 1 is capable of processing transaction messages to take over the load.

back to top

9.2.10 URID Replication

Within each IBM Safer Payments instance, each transaction record stored is assigned a unique record Id (URID). URIDs are assigned incrementally; the first record thus has an URID of zero. Since URID counting is never restarted, each transaction ever processed by IBM Safer Payments is uniquely identifiable.

This is however more complex in a clustered IBM Safer Payments installation. Because each IBM Safer Payments instance must be able to operate autonomously, it is impossible for IBM Safer Payments instances to negotiate URID values before they store a transaction record. Thus URID are only unique within an IBM Safer Payments instance. If the authorisation system uses three IBM Safer Payments instance in a round-robin scheme, the first transaction record would be URID=0 on instance 1, the second URID=0 on instance 2, and so forth.

Hence in a cluster, a transaction record is only uniquely identified by the instance Id and URID of the ("primary") IBM Safer Payments instance that initially received the transaction message (or processed the batch data record). Therefore two attributes "Primary instance ID" and "Primary URID" must be created with each model revision as meta attributes in a clustered configuration. Each primary IBM Safer Payments instance then adds the primary URID and its instance Id to the FLI message passed to the other IBM Safer Payments instances, where they get stored with the transaction record in addition to the ("secondary") URID of this specific IBM Safer Payments instance.

The unique identification of transaction records is needed both for auditing purposes and also with the flagging of transactions as fraud during investigation. In the latter case, the primary URID and instance Id values are used to ensure that the same transaction is flagged (or un-flagged) as fraud on all IBM Safer Payments instance in a cluster.

back to top

9.3 Remote Operation of IBM Safer Payments

While all operational processes and maintenance tasks are supported from within IBM Safer Payments, they can also be operated remotely from any central datacenter environment using the IBM Safer Payments API (Application Programming Interface).

The remainder of this page exemplifies the use of the API for typical operational processes and maintenance tasks. Notice that the operations can be executed on all IBM Safer Payments server instances, regardless on whether or not their API is set active. Contact the IBM Safer Payments support for a complete API reference manual.

Ping

The API ping can be used on any IBM Safer Payments server instance to test if the instance is responsive. No valid user session (login) is required for this. Load balancing / failover switching gear between the IBM Safer Payments server instances and user access can use this API request to determine the API active IBM Safer Payments server instance.

Login/logout

All API requests (other than "ping") require authentication. Typically, a specific user account is created for such remote operations.

The user account should be created with the "enforce password changes" option disabled. Once the "login" request is responded to, the session remains active for the specified timeout period. The session cookie must be retrieved from the response, as it must be sent back to the API with all subsequent API requests.

Cluster status

A complete status overview can be obtained with the "getInstancesTable" request.

Switching active API

To enable the API on an instance (and automatically disable it on all other), send the "enableApi" request.

Taking an instance offline (e.g. for backup)

To disable/close all interfaces on an IBM Safer Payments server instance instance (for instance before executing a shutdown or to isolate an instance for non-shadow copy enabled backup), send the "setOffline" request.

This request disables/closes the MCI, API, BDI, and incoming FLI interfaces of the respective IBM Safer Payments server instance. Notice that you must first activate the API on another IBM Safer Payments server instance for users to still have access to the IBM Safer Payments cluster (user sessions lost during switching active API). The SCI interface remains enabled/open since it is used by the IBM Safer Payments cluster instances to exchange necessary control and status information even with an offline IBM Safer Payments server instance. Since the operating SCI does not change any file stored data, it can safely be kept open for instance during a backup of the instance.

Taking an instance online (e.g. after a backup)

To re-activate all enabled interfaces after having taken it offline, send the "setOnline" request.

Shutdown

To initiate the shutdown of an instance (and automatically disable it on all other), send the "shutdown" request.

Notice that you do not need to shut down an IBM Safer Payments server instance for certain maintenance tasks, such as backup without shadow copies: taking the instance offline is sufficient.

back to top

10. Appendix

This chapter contains various reference sections.

back to top

10.1 IBM Safer Payments Architecture and Integration

This section introduces various aspects of IBM Safer Payments architecture and IBM Safer Payments integration.

back to top

10.1.1 Interfaces Overview

The IBM Safer Payments service provides multiple Interfaces:

- MCI (Message and Command Interface) *real-time*
- API (Application Programming Interface) *user access*
- BDI (Batch Data Interface) *files*
- SCI (Status and Control Interface) *cluster control*
- ECI (Encrypted Communication Interface) *exchanging secrets*
- FLI (FastLink Interface) *redundancy*
- RDI (Relational Database Interface) *database*
- AMI (Alert Message Interface) *outgoing channels*
- MQI (WebSphere MQ Interface) *message queueing*

While MCI, API, SCI, FLI, AMI, and MQI are IP message based message interfaces, BDI and RDI interfaces are file based for batch data.

The MCI, API and FLI interfaces operate in "service mode", where each communication is initiated by the outside party and IBM Safer Payments replies to each request. With these interfaces, the IP connections typically stay open for more than one request (for reasons of efficiency). This rather simple communication scheme keeps interfacing to IBM Safer Payments easy. It follows the time tested model of most Internet protocols, where the service consumer (often a browser) polls data from the service provider (often an HTTP server) whenever it needs to. For performance reasons, all three IP based interfaces use thread pool technology.

The BDI interface is quite different from the others because it involves transferring data in and out of IBM Safer Payments via files. Because this requires IBM Safer Payments to become active at specific times to check if data to be imported is available or if data should be delivered to other systems, IBM Safer Payments features a job schedule function.

While MCI and BDI are "external" interfaces in the sense that they connect IBM Safer Payments to systems of the customer, API and FLI are "internal" interfaces in the sense that they connect IBM Safer Payments components. The API connects the IBM Safer Payments client and the IBM Safer Payments server, the FLI connects different IBM Safer Payments instances within a cluster.

The RDI is a batch file interface using SQL statements to transfer IBM Safer Payments data into a relational database.

The AMI uses outgoing channels to send messages or queries to users, administrators, customers, and cardholders/merchants. It currently supports sending messages to SMTP (email), HTTP, IP, ODBC (database), and file targets.

The MQI allows to connect to existing IBM WebSphere MQ message queueing environments and read queue message data.

All interfaces are described in detail on separate help pages that opens when you click on the respective interface hyperlink above.

back to top

10.1.1.1 Doublet Detection

This feature allows IBM Safer Payments to detect transaction messages (received by both MCI and API) that have already been stored in IBM Safer Payments' MDC/DDC and to **not** store them again.

Configuration

How IBM Safer Payments detects doublets (once enabled), is configured be the following settings:

- Include DDC
  If enabled, also data from DDC is used to compare the current transaction message with (may substantially slow down real-time performance).
- Index
  To deliver meaningful performance, doublet detection is only performed alongside an index. Choose the index that has the largest number of entries for best performance. Typically this would be an index for the cardholder as there are typically less transactions to check as potential doublets for each cardholder than for each merchant or terminal.
- Attributes
  Select all attributes that IBM Safer Payments shall compare to determine if two transactions are the same (it is implicit that selected index value must be the same).

Notice that doublet detection reduces run-time performance.

Behavior

If doublet detection is enabled, a detected doublet message will not be stored in MDC/DDC and will not be computed (that is, calendar profiles/events are not updated), and the response (sent back with MCI transaction messages and stored in the .log files with BDI transaction messages) contains the error message "Doublet detected, message discarded" and generates the event log message 284.

back to top

10.1.1.2 Message and Command Interface (MCI) Overview

The message and command interface ("MCI") connects IBM Safer Payments to authorization systems, card management systems and related data sources (aka service consumer).

The MCI uses TCP/IP message passing, either as "naked" XML messages or XML messages wrapped in HTTP. While in both cases, you may open a connection, send the request, wait for the response and close the connection immediately thereafter, we highly recommend keeping the connection open between request/response pairs, since such "persistent connections" allow for a much higher transaction message throughput and lower resource consumption.

For messages wrapped into HTTP the following header fields are neccessary

- Content-Length (length of the message included in the request)
- X-SP-Message-Type-Id (messageTypeId which represents the message type defined in SP and will be used to identify the parser to interpret the message)
- X-SP-Message-Id (The MessageId is an alphanumerical string value that is generated by the systems connected to IBM Safer Payments to identify a response. (Identifying messages can be useful if multiple connections to IBM Safer Payments are used in parallel.) IBM Safer Payments does not use this response for any computational purposes; it only echoes it back in its response (it also uses the MessageId in case of errors to document the offending message). The "MessageId" may contain up to 16 Bytes and consist of any ASCII character between 32 and 126, with the exception of ">", "/", "<" characters.)
- X-SP-Protocol-Version (protocol version of SP, currently it is fixed to 1)
- X-SP-Request-Type (optional) defines what kind of request is send ((default)ModelRequest / StatusRequest)

For messages sent via IP a binary header is needed, which consists of 64 Bytes and includes the same header information as for HTTP.

- first 8 bytes define the Content-Length
- next 4 bytes define the X-SP-Message-Type-Id
- next 16 bytes define the X-SP-Message-Id
- the next byte defines the X-SP-Protocol-Version
- the next byte defines the X-SP-Request-Type (0 for ModelRequest/ 1 for StatusRequest)
- the last 34 bytes are reserved for future use

Message examples

This is an example of an XML request message sent by the service consumer to IBM Safer Payments:

```
<IRIS><PAN>1234567890123456</PAN><TrxDateTime>2014-04-22 21:04:56</TrxDateTime><Amount>123.45</Amount>
<MCC>5512</MCC><CC>BE</CC></IRIS>
```

An example XML response message by IBM Safer Payments would be:

```
<IRIS Version="1" Message="ModelResponse" MessageId="000af87c75503b4401" ErrorCode="0"><Intercept>1</Intercept>
</IRIS>
```

This is an example of an HTTP POST wrapped XML request message sent by the service consumer to IBM Safer Payments:

```
POST /path/script.cgi HTTP/1.0
From: test@mybank.com
User-Agent: HTTPTool/1.0
Content-Type: text/xml
Content-Length: 188
X-SP-Message-Type-Id: 101
X-SP-Message-Id: 000af87c75503b4401
X-SP-Protocol-Version: 1
X-SP-Request-Type: ModelRequest

<IRIS><PAN>1234567890123456</PAN><TrxDateTime>2014-04-22 21:04:56</TrxDateTime><Amount>123.45</Amount>
<MCC>5512</MCC><CC>BE</CC></IRIS>
```

An example HTTP POST wrapped XML response message by IBM Safer Payments would be:

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/xml; charset=utf-8
Content-Length: 275

<IRIS Version="1" Message="ModelResponse" MessageId="000af87c75503b4401" ErrorCode="0"><Intercept>1</Intercept>
</IRIS>
```

Notice that IBM Safer Payments auto-detects formats, if your response contains an HTTP POST wrapper, it will be responded to with an HTTP response wrapper.

Connections

In a typical IBM Safer Payments application, the service consumer will open multiple connections at once. The maximum number of connections that can be opened with IBM Safer Payments at the same time is defined by the threadpool size setting on the IBM Safer Payments system configuration page. Because IBM Safer Payments uses a separate computational service thread for each connection, using multiple connections in parallel increases the transaction throughput of IBM Safer Payments.

Notice that while within each connection, all transaction request messages are processed and responded to in exactly the sequence they arrive (synchronous), while there is no guaranteed sequence of processing and responding between multiple connections (asynchronous).

Status request

To test if the MCI is active on a specific port, you can send the request:

```
<IRIS Version="1" Message="StatusRequest"></IRIS>
```

which will be responded with:

```
<IRIS Version="1" Message="StatusResponse" InstanceStatus="Ok" ></IRIS>
```

Notice that this request can also be sent wrapped as HTTP POST.

Error handling

In normal operations, once IBM Safer Payments has started up, it opens a listener socket on the port defined for the MCI (on the cluster page). For each connection opened by the service consumer, IBM Safer Payments uses one of its pooled computational threads to serve the transaction message requests. If the service consumer does not need IBM Safer Payments any more, it can close a connection, and later, when necessary, re-open it. Typically the service consumer keeps the connection open for its entire uptime, if no error occurs. In normal operating conditions, IBM Safer Payments will never close a connection.

There are, however, certain error conditions in which IBM Safer Payments closes a connection. For instance if a message is so malformed that IBM Safer Payments cannot detect its end, it must close the connection because it cannot know when the next message starts and can thus not parse it. Also if a FastLink buffer is filled above a critical threshold (defined on systems configuration page), IBM Safer Payments drops its MCI connection to avoid data inconsistency within a cluster, thereby forcing the service consumer to switch to the next IBM Safer Payments instance of the cluster. If IBM Safer Payments experiences a severe operational problem and shuts down, it will also close all open MCI connections.

Redundancy and load balancing

In an IBM Safer Payments cluster, under normal operating conditions, all IBM Safer Payments instances accept transaction messages via their MCI. The service consumer is thus completely free in its decision to which instance to send its data to. The IBM Safer Payments instances of a cluster replicate data internally.

In order to ensure redundancy, the service consumer must use a watchdog mechanism for each transaction message request sent. If IBM Safer Payments has not replied to request within the allotted time (or closes the connection), the service consumer must re-send the message to the next available IBM Safer Payments instance.

Notice that because each IBM Safer Payments instance must completely compute a transaction message whether it has been the one instance the service consumer has sent the message to ("primary" IBM Safer Payments), or whether it had received this message from another IBM Safer Payments instance as replication (via FastLink), very little can be gained from using any load balancing techniques when sending transaction message requests to IBM Safer Payments.

XML format

It is important to notice that IBM Safer Payments' MCI supports only a subset of the XML format (for performance reasons). Thus MCI requests must exactly follow the format laid out in this section to avoid request interpretation errors. The XML format rules are:

- Never use whitespaces in messages. No spaces between elements or any other "filler" characters (such as tabs), carriage returns or line feeds. If the IBM Safer Payments XML data format calls for space characters – for instance between an element name and an attribute name – you must use exactly this one space character. If the alias name used in the mapping definition of a message contains space characters, they may be present in the respective XML element.
- Only use standard quote characters (ASCII value 34), no single quotes, no italic quotes.
- The XML format is case sensitive.
- If an attribute value of your request message is empty, do not use <aliasName/>. Use <aliasName></aliasName> or skip this attribute entirely from the message.
- Only use characters in the ASCII range from 32 to 126 or UTF-8 encoded values.
- CDATA sections are *not* supported
- Predeclared entities &amp; &lt; &gt; &apos; &quot; are supported.

All requests are contained within <IRIS> elements so that their start and end can be identified with the open connection data stream (persistent connection). This is an example of a standard IBM Safer Payments request transaction message:

```
<IRIS Version="..." Message="..." MessageTypeId="..." MessageId="...">...</IRIS>
```

Mandatory attributes in the opening <IRIS> element or in the header with any request are:

- Version (X-SP-Protocol-Version)
  The "Version" attribute is used to differentiate different sub-types of request messages. Version="1" is the standard IRIS XML message version, other version numbers are used for customer specific implementations.
- Message (X-SP-Request-Type)
  Identifies the type of message:
  - "ModelRequest" sends data from the service consumer to IBM Safer Payments for computation.
  - "StatusRequest" asks IBM Safer Payments for basic health parameters and can also be used to "ping" if an MCI port (or the IBM Safer Payments instance serving it is responsive).

For "ModelRequest" requests, the following attributes must be present in the opening <IRIS> element or in the header of the message:

- MessageTypeId (X-SP-Message-Type-Id)
  The "MessageTypeId" is the (numeric) value used by IBM Safer Payments to identify the type of transaction message (messages are created and maintained in their own section on the administration tab, see the IBM Safer Payments online help for messages for details).
- MessageId (X-SP-Message-Id)
  The MessageId is an alphanumerical string value that is generated by the systems connected to IBM Safer Payments to identify a response. (Identifying messages can be useful if multiple connections to IBM Safer Payments are used in parallel.) IBM Safer Payments does not use this response for any computational purposes; it only echoes it back in its response (it also uses the MessageId in case of errors to document the offending message). The "MessageId" may contain any ASCII character between 32 and 126, with the exception of ">", "/", "<" characters.

For "StatusRequest" requests, there are no other attributes in the opening <IRIS> element. The "StatusRequest" request hence is always like this:

```
<IRIS Version="1" Message="StatusRequest"></IRIS>
```

No specific sequence of the attributes listed above within the <IRIS> element is required. Attributes provided in a request but not listed above are ignored by IBM Safer Payments. Notice that in future releases, we may add attributes and change their sequence.

Responses are also contained within <IRIS> elements. Which attributes are present in the opening IRIS XML element is defined on the

"IBM Safer Payments System Configuration" page. This is an example of a "ModelResponse":

```
<IRIS Version="..." Message="ModelResponse" IrisInstance="..." MessageTypeId="..." SystemTime="..." UniqueRecordId="..."
"MessageId="..." Mandator_n="..." Revision_n="..." Merging="..." InstanceStatus="..." Error="..." ErrorCode="...">...</IRIS>
```

If the value of "Respond max fired rules" is defined as non-zero value (on the "IBM Safer Payments System Configuration" page), the attribute value elements are rather contained within the <Output> elements. Here the <RulesFired> element contains the attribute NumRulesFired that returns the number of rules that actually fired. Between the opening and closing <RulesFired> elements, a <Rule> element for each fired rule is provided. Here is an example:

```
<Rule><Mandator>...</Mandator><Revision>...</Revision><RuleID>...</RuleID><RuleName>...</RuleName>
<RuleComment>...</RuleComment><RuleSetPriority>...</RuleSetPriority><RulePriority>...</RulePriority></Rule>
```

This is an example of a "StatusResponse":

```
<IRIS Version="..." Message="StatusResponse" InstanceStatus="..." >...</IRIS>
```

The opening <IRIS> element of a response (depending on the settings on the "IBM Safer Payments System Configuration" page) may contain the attributes:

- Message
  Type of message:
  - "ModelResponse" contains IBM Safer Payments computed result for the model request.
  - "StatusResponse" is what IBM Safer Payments sends back to a "StatusRequest".
- IrisInstance
  The name of the IBM Safer Payments instance that has computed the response as defined in the cluster settings.
- MessageTypeId
  Echoed from "ModelRequest".
- SystemTime
  System timestamp of the IBM Safer Payments instance that has computed the response (taken from the IBM Safer Payments instance hosting server) in the format "YYYY-MM-DD hh:mm:ss".
- UniqueRecordId
  This is the internal unique indentifier of the (primary) IBM Safer Payments instance that has responded to the request. It is delivered with all responses since there are IBM Safer Payments applications where the service consumer sending the request needs to keep the exact reference of the transaction message as stored in IBM Safer Payments. Notice that the value is "-1" if the transaction message request did not result in the creation of a transaction record in IBM Safer Payments (e.g. fraud alerts or postings that merged to existing records, or transaction messages delivering customer standing data (masterdata)). This attribute is of numeric type and in the range from 0 to 9,223,372,036,854,775,808.
- MessageId
  Echoed from "ModelRequest".
- Mandator_n / Revision_n
  These pairs of attributes are contained for each mandator model that participated in the computation of the response. The numbers n are simple roll numbers, the Mandator_n mandator name is provided as the name of the mandator as defined in IBM Safer Payments, the respective champion revision is provided as the revision number rnum. Both values can be used by the service consumer if it needs to exactly record which decision model revisions had an influence in the computation of the response.
- Merging
  If the request was a merging source, Merging="1" is returned.
- InstanceStatus
  Status of this instance. Possible values are: "Ok", "StartingServices", "Error", "Startup", "RestoringDonor", "RestoringRecipient", "Invalidated", "Unreachable", "Undetermined", "Lockdown", "WaitingForSynchronization", "Synchronizing", "WaitingForKey", "RestoreFailed", "Reencryption", "Golive", "ShutdownRequested", "Dumping". Only in status "Ok" and "ShutdownRequested" IBM Safer Payments processes ModelRequests. Status "ShutdownRequested" indicates a situation in which the service consumer should switch over to another IBM Safer Payments cluster instance as this one will soon become unavailable.
- Error/ErrorCode
  If the value of "ErrorCode" is non-zero (and the text value of "Error" is not empty), IBM Safer Payments has encountered an error situation that was not severe enough so that IBM Safer Payments had to actually close the connection (see above; for instance, if because of malformattings, IBM Safer Payments is unsure where the current transaction message ends and the next one begins, it must close the connection). Errors are listed in the table below:

ErrorCode	Error	Comment
1000	No message defined for MessageTypeId=n	
1001	Doublet detected, message discarded	Only available if doubled detection is enabled in IBM Safer Payments
1002	Parsing error in transaction message 'IRIS' element	
1003	Incoming transaction message of unknown type	
1004	Parsing error in transaction message variables	
1005	The attribute 'MessageTypeId' of the XML IRIS element is missing from request	
1099	*Error message passed through*	

Notice that in future releases, we may add attributes and change their sequence.

XML value formats

IBM Safer Payments supports different data types for which the following formatting rules apply:

- Numeric
  All numeric values may or not contain the period (".") as decimal separator. Negative values are preceeded with the minus ("-").

No exponential format, no currency character(s), and no digit group separators may be present. The number of decimals used from the values delivered, and the minimum and maximum value depend on the settings of the respective attribute in the model. Values higher or lower than the limits are clipped to the respective limit.

- Text
  All characters between the opening and closing variable element is considered text value. The characters &, <, >, ' and " must be escaped using predeclared entities &amp; &lt; &gt; &apos; &quot;. Characters must also be in the ASCII range from 32 to 126 or 128 to 255. Values longer than defined with the respective model attribute are clipped to this length.
- Timestamp
  All timestamp values must be ISO-formatted, that is follow exactly the "YYYY-MM-DD hh:mm:ss" format ("YYYY": year in four digits, "MM": month in two digits, "DD": day in two digits, "hh": 24-hour in two digits, "mm": minutes in two digits, and "ss": seconds in two digits). If you need to enter a date as a timestamp, add " 12:00:00" to the ISO date format. Milliseconds are not supported.
- Time interval
  All time interval values use a format similar to ISO 8601 for each side of the interval and the "~" symbol to separate them. Several types of date and time information are supported but both sides of the interval must use the same type:
  - Full timestamps: Both sides are formatted exactly as you would when sending just a single timestamp. Example: "2018-03-01 09:00:00~2018-03-15 12:00:00".
  - Time only: Both sides consist of a time only. Possible formats are "hh:mm:ss", "h:mm:ss", "hh:mm" and "h:mm".
  - Day of the week only: Both sides use the english names of the weekdays either in full or abbreviated. Case does not matter. Example: "mon~wed".
  - Day of the week with time: Combines the type above with a time. All time formats mentioned above can be used. Example: "mon 9:00~wed 12:00:00".
  - Day of month and mont only: Both sides appear similar to full timestamps but with a "-" instead of the year and without a time. Example: "--03-01~--03-15".
  - Day of month and month with time: Combine the type above with a time. All time formats mentioned above can be used. Example: "--03-01 9:00~--03-15 12:00:00".
  - Day of month only: Both sides appear similar to full timestamps but with a "-" instead of the year and the month and without a time. Example: "----01~----15".
  - Day of month with time: Combines the type above with a time. All time formats mentioned above can be used. Example: "----01 9:00~----15 12:00:00".

JSON Format

The MCI allows sending messages in JSON format.

- Because array elements are referenced in the mapping with "[number]", "[" and "]" are forbidden characters in element names.
- If the alias name used in the mapping definition of a message contains space characters, they may be present in the respective XML element.

Example of a JSON message request:

```
{"Message": {"Amount": 5497558138.88,"Payer": {"PAN": 36028797018963968},"Array": ["Value1","Value2"],"Timestamp":
"YYYY-MM-DD hh:mm:ss"}}
```

Nested elements can be accessed by writing the path into the alias in mappings, where "/" seperates the keys. For example to access the element Amount you need to set Message/Amount as mapping alias.To access the second element in the array, it would be "Message/Array[1]".

Example of a JSON message response:

```
{
"IRIS": {
"Version": 1,
"Message": "ModelResponse",
"IrisInstance": "InstanceName",
"MessageTypeId": 11,
"SystemTime": "2014-04-22 21:05:18",
"UniqueRecordId": 123456,
"MessageId": "0af87c75503b4401",
"Mandator": [
{
"Mandatorname": "MyBank",
"Revision": 144
}
],
"Merging": true,
"InstanceStatus": "Ok",
"Latency": 0.71,
"Error": "",
"ErrorCode": 0
},
"Outputs": {
"transactionOut": {
"computedOut": {
"values": [
"aaaa",
"aaaa"
]
},
"hexValue": "xxxxxxxxxxxxxxxx"
}
},
"RulesFired": {
"NumRulesFired": 1,
"Rules": [
{
"Mandator": "MyBank",
"Revision": 144,
"RuleID": 2014,
"RuleName": "ATM Skimming",
"RuleComment": "In connection with skimmed card data from ATMs.",
"RuleSetPriority": 1000.0,
"RulePriority": 750
}
]
}
}
```

The response has the same information as for XML responses. All output attributes mapped for the message are written in the element "Outputs". You can also use the mapping to nest all outputs, with the same syntax, which is used for incoming messages. If array elements are skipped, for example only output attribute with mapping array[3] is enabled for the message, then positions 0 to 2 are filled with empty strings.

HTTP capabilities

The MCI also supports transaction message requests as HTTP POST requests. While persistent connections (as defined in HTTP 1.1) are supported, chunked transfer-encoding is not supported. Notice that none of the POST header values including the path value will be evaluated by IBM Safer Payments. IBM Safer Payments only evaluates the request's contents enclosed with the <IRIS> elements.

Direction

The MCI interface is an incoming interface, each connection is initiated by the service consumer and also terminated by it.

Custom Messages

The MCI allows sending custom messages and receiving custom responses. To send custom messages through the MCI interface, you would need to define a custom header in the HTTP request with the key being "parser" and value "custom". You would also need to configure the MCI configuration settings accordingly and make sure you have the custom parser library in place. The default XML format only supports a single layer of elements under the IRIS tag, with no support for nested elements.

Notice that further information on the MCI is provided at the IBM Safer Payments support.

back to top

10.1.1.3 Application Programming Interface Overview

The Application Programming Interface (aka "user access interface") connects the IBM Safer Payments server component to the IBM Safer Payments client component used by end users (via browser software) and to third-party software components (the term "Third-party components" in this context is used to summarize any other software component that accesses or extends IBM Safer Payments functionality. These components are in the sense optional that IBM Safer Payments in itself contains all standard functionality for fraud prevention applications. Many IBM Safer Payments users, however, like to expand, customize or integrate IBM Safer Payments with their systems).

Cluster installation

Since IBM Safer Payments is typically installed in a clustered environment involving multiple IBM Safer Payments server instance nodes, the connection from the IBM Safer Payments clients (running in the users' browsers) to the cluster can be facilitated in different ways.

In an IBM Safer Payments server cluster, only one instance serves user requests at a time ("active API"). The API of all other instances of the cluster remains deactivated. This is necessary as otherwise multiple users logged on to different instances could change the same data object in a non-consistent way.

As long as all instances are in synchronization, the administrator can freely switch the API activation between IBM Safer Payments server instances from the cluster settings page or remotely from scripts. Notice that when the API is activated on another instance -- deactivating the API on the current instance -- all user sessions are terminated as a consequence of this, and all unsaved data of users is lost.

Notice that all instances with deactivated API are still accessible via its API for certain administrational tasks (in particular, cluster administration). This ensures that if IBM Safer Payments server instance with the active API becomes unavailable, the administrators can use any other IBM Safer Payments server instance to switch the active API.

- Simple implementation
  The most simple implementation of user access in a cluster installation thus is to have IBM Safer Payments users use one "dedicated" IBM Safer Payments server instance that has its API activated. In case of failure, update, or any other kind of

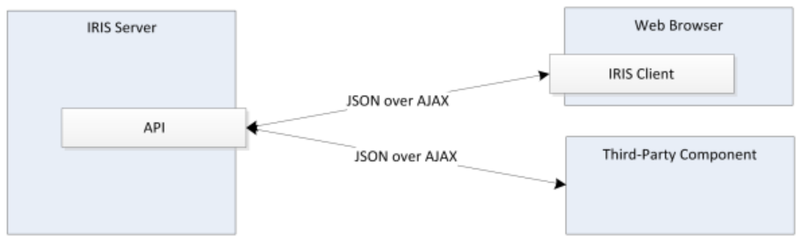
scheduled maintenance that requires this instance to be taken down, if users shall be able to continue their work, the API must be activated on another IBM Safer Payments server instance, and the users must resume working on this instance. For IBM Safer Payments cluster installations with very few users (for example, installations not using the case investigation, analysis, and reporting capability of IBM Safer Payments), or installations in which the user access is not 24/7, this approach often is sufficient.

- Failover implementation
  With larger number of users, telling them to use different servers becomes a burden. In this case, you may use an external fail-over switch to route user requests automatically to the active API IBM Safer Payments instance. This makes all IBM Safer Payments instances API ports visible under the same IP address and ports for all users. If the administrator now switches the active API to another instance, users can continue their work after they logged in again. The administrator intervention can be from within IBM Safer Payments (administration/cluster page), or remotely manually or automatically.

IBM Safer Payments client

The API uses a subset of HTTP (or HTTPS) as transfer protocol: URL encoded HTTP requests to send information from the browser to the IBM Safer Payments server as AJAX, and JSON formatted data for the response sent back from the IBM Safer Payments server to the browser.

The IBM Safer Payments client component is implemented as a set of JavaScript libraries that runs entirely within each user PC's web browser (the libraries are automatically loaded (and cached) by the web browser from the IBM Safer Payments server on an as-needed basis). Its internal architecture follows the MVC (Model View Controller) approach that decouples data from its representation. Therefore the communication between IBM Safer Payments server and client is strictly data based (simply put: no HTML). The MVC sends AJAX requests to IBM Safer Payments describing the action that needs to be performed by IBM Safer Payments or the data needed and receives back status information and data in JSON format.



The same technology is used by third-party software components to access IBM Safer Payments functions and data.

To serve the needs of both the IBM Safer Payments client and third-party components, the IBM Safer Payments API comprises a superset of API calls (AJAX request types). The API does not differentiate between functions for the IBM Safer Payments client and third-party components, thus third party components may access the full IBM Safer Payments client functionality. The IBM Safer Payments API hence is comprehensive and for instance would even allow for third parties to write a completely different user access interface.

In addition to the IBM Safer Payments API requests, the IBM Safer Payments client also requests a number of static files from IBM Safer Payments (these static files are contained in the "inc" subdirectory of the IBM Safer Payments installation), including JavaScript libraries, CSS and image files.

Because of the HTTP transport layer used by the IBM Safer Payments API, the IBM Safer Payments server typically is configured to listen to requests on the HTTP port that by default is 80.

Request/response formats

The IBM Safer Payments API is based on AJAX technology. AJAX typically refers to a combination of techniques, including the use of JavaScript on the browser side, HTTP as transport layer/protocol and XML as format for the responses. The IBM Safer Payments client and API implementation differs from this in one respect. Responses from IBM Safer Payments are not XML, but JSON formatted. The primary reason for it is that JSON is more succinct than XML, however, since both formats can be transferred easily back and forth, the choice of JSON over XML for the AJAX responses has no practical consequences.

API port ping

To test if the API is active on a specific port, the request:

```
iris.mybank.com?{"request":"ping"}
```

can be sent to the API. If the API is active, this request is responded with:

```
{"responseStatus":["OK","API_ACTIVE"],"reloadUserProfile":false}
```

even when the requesting party has not established a valid session. Notice that it is not necessary or recommended to ping the API port. This functionality is provided only for testing and for certain load balancing / failover equipment.

Requests

IBM Safer Payments API requests have one JSON object after the IBM Safer Payments installation URL. Here is an example:

```
iris.mybank.com?{"request":"save","uid":-1,"type":"","mandator":1033,"revision":1051,"ruleset":-1,"data":{}}
```

Of these JSON variables, only the first one "request" is mandatory. If the other variables must be provided or not, depends on the type of request. If they are provided, they must be provided in this sequence:

- request
  This variable identifies the request type as text value. It must always be present for IBM Safer Payments to understand what it is

asked to do.

- uid
  Many requests reference an IBM Safer Payments element identified by a UID. For these requests, the UID is provided with this JSON variable. For requests that save settings of an IBM Safer Payments element, the value "-1" indicates that this IBM Safer Payments element has not yet been created in IBM Safer Payments and therefore must first be created before saved. IBM Safer Payments in this case generates a new unique ID (UID) for it.
- type
  Some requests exist in variants. In this case, the type variable denotes this variant.
- mandator
  For all requests that target mandator specific actions, this variable transmits the UID of the respective mandator.
- revision
  For all requests that target model revision specific actions, this variable transmits the UID of the respective revision.
- ruleset
  For all requests that target ruleset specific actions, this variable transmits the UID of the respective ruleset.
- data
  Some requests, mostly the ones that save an entire IBM Safer Payments element, need to deliver structured data to the IBM Safer Payments server. In this case, the data variable delivers a JSON object with this data.

The IBM Safer Payments API supports only GET type requests. There must not be space or other "filler" characters outside text values in quotes.

Responses

All responses from IBM Safer Payments to the browser are JSON formatted. All JSON variable names in IBM Safer Payments start with small caps and – if name is combined – use camel case thereafter. They all first contain the (optional) actual responseData followed by the variable "responseStatus" describing the status of the response:

```
{ responseData, "responseStatus": ["status", "feedbacktext"], "reloadUserProfile" : true|false, "csrfToken": csrfToken }
```

With the variable "responseStatus", an array containing one or two (first one mandatory) values, depending on the response status:

- OK with optional feedback
  status is "OK" and *feedbacktext* may contain informational feedback that can be shown to a user (if the request was from a UI) or a log file (if the request was from a third party component).
- Warning
  status is "W" and *feedbacktext* contains an warning feedback that can be shown to a user (if the request was from a UI) or a log file (if the request was from a third party component).
- Error
  status is "E" and *feedbacktext* contains an error feedback that can be shown to a user (if the request was from a UI) or a log file (if the request was from a third party component). Such an error should be alerted to the user with a modal dialog box.
- Fatal
  status is "F" and *feedbacktext* contains an error feedback that can be shown to a user (if the request was from a UI) or a log file (if the request was from a third party component). The difference to the "Error" is that "Fatal" is for errors that are assumed to be not correctable by the user. They typically are assumed internal software (UI-service) mismatches or the result of improper API requests (that could for instance also be the result of a user manipulating HTTP requests). Because "Fatals" are not expected to ever be shown to a user, they are in English language and not translated into any language.
- Session Expired
  status is "SEX" and *feedbacktext* contains an informational feedback that can be shown to a user (if the request was from a UI) or a log file (if the request was from a third party component).

With many IBM Safer Payments API requests, there is no responseData, leaving "responseStatus" the only returned variable.

The "csrfToken" is provided to protect IBM Safer Payments against "cross site request forgery" (constant for a session).

Sessions

All API access, whether by users or by third-party components, is granted only within a valid IBM Safer Payments session and thus require a user account to be associated with (Because the JavaScript code in the browser cannot be fully protected against manipulation, the session Id is stored and checked to be valid on the IBM Safer Payments server, where it cannot be manipulated). In addition, all API requests are subject to the user/group privilege model of IBM Safer Payments. Thus the first step using API requests is to initiate a session with the IBM Safer Payments server.

How this is done differs a bit between user access and third-party component access. User access requires one first step before because users are able to access the IBM Safer Payments server from a standard web browser with no additional software installed.

Typical web sites use an HTTP server to deliver HTML pages to the browser upon its requests. These HTML pages may be static or server-generated. IBM Safer Payments uses a radically different approach.

When users access IBM Safer Payments via the installation URL, the embedded HTTP service function of IBM Safer Payments delivers an HTML page to the browser that only contains links to JavaScript libraries (In other words, the HTML page only contains a header with the links to the JS libraries and the body contains the call to the JS library main function. It is – with the exception of the links – empty). These libraries contain the entire user access component of IBM Safer Payments that from that moment on run user access to IBM Safer Payments. No other HTML page is loaded after this.

Once the JS libraries are loaded (depending on the environment, this process typically takes less than a second), the IBM Safer Payments MVC (model view controller) in the web browser is started. Detecting that no session is valid; the controller first invokes the log-in process. Then, depending on the user actions, the controller moderates the communication between the browser and IBM Safer

Payments via AJAX requests. For third-party component IBM Safer Payments API access, a session starts directly with the invocation of the login process.

The API identifies a session through a browser cookie. The response to a successful login request contains an HTTP cookie that is stored with the browser. For third-party component access, this cookie is set for instance by the line:

```
Set-Cookie: sessionIdn=sessionId;path=/; HttpOnly
```

contained in the HTTP header of the IBM Safer Payments response.

Notice *n* corresponds to the instance ID of the respective IBM Safer Payments instance (as defined in the "id=*n*" command line parameter). This enables multiple sessions with different IBM Safer Payments instances on the same browser.

Now every request from the browser or the third-party component must send the sessionId as cookie in its HTTP request header, for instance by including the line:

```
Cookie: sessionIdn=sessionId
```

in its header. The session ID value remains the same during the session.

If enabled in its settings, IBM Safer Payments uses a second session token to prevent CSRF (cross-site request forgery) attacks. This token is submitted as part of each API response as JSON variable "csrfToken" explained above. Its value must be passed back with each subsequent request as the HTTP header element:

```
CsrfToken: csrfToken
```

Remarks

- The sequence of JSON request objects and variables must exactly follow the sequence specified. There may be objects/variables missing (if the default value applied would work), and there may be additional objects/variables (which will be ignored), but the defined objects/variables must always be provided in the specified sequence.
- Notice that whitespaces and linefeeds are added in this documentation for readability with any printed JSON example. The actual responses of the IBM Safer Payments API may not have these for reasons of efficiency.
- The "reloadUserProfile"=true variable indicates if an administration change impacts the current user session. If this variable is present, the user profile – if cached – should be updated from IBM Safer Payments to avoid errors from actions for which no privileges exist.

A full reference on all API requests and responses can be obtained from the IBM Safer Payments support.

Multi-user capabilities

IBM Safer Payments is designed for a large number of concurrent users. It thus keeps locks for viewed-/edited contents at a minimum to ensure maximum productivity. Different lock mechanisms are used for different parts of IBM Safer Payments:

- Administrative functions
  Most administration pages have a table that lists the number of existing items. Clicking on an entity opens an entry form below that lets the user view the properties of the respective item. Users with edit privileges for this item in addition get a "New XYZ" button above the table to open the entry form empty for the creation of a new item, and when they click on an existing item, the form opens editable and "Save"/"Delete" buttons are presented. To avoid that multiple users edit the same item, the first user that opens an item in editable form implicitly reserves the item (the "reservation" is kept until the "lock timeout" period, as defined in this section, is not expired). Each other user that clicks on the item in the table will only get the form in non-editable mode. Because with all items the item names must be unique, the uniqueness of the item name is checked right within the JavaScript code in the form. However, because multiple users with edit privileges can simultaneously create new items, the unlikely case that two users simultaneously create two items with the same name, there is a second check for name doublets when the item properties are actually saved.
- Decision model maintenance
  Locking of model revisions is described on the online help page of the revision selection on the "model" tab page.
- Case investigation
  Locking of cases is described on the online help page of the "investigation" tab page.

Direction

The API interface is an incoming interface, each connection is initiated and terminated by the web browser, its (reverse) proxy, or a third-party application.

back to top

10.1.1.4 Batch Data Interface Overview

The batch data interface ("BDI") is used for various purposes in IBM Safer Payments. In essence, it is a vehicle to get data in and out of IBM Safer Payments as data files. The two primary uses of this interface are:

- Ad hoc
- Periodical

Typical ad hoc use cases are to get analysis data into IBM Safer Payments, for example as part of a feasibility study or offline model generation, or to extract data from IBM Safer Payments for use in other systems. The job scheduling page (administration tab) provides a one-glance overview on all jobs scheduled and their status.

Data format

IBM Safer Payments supports multiple formats for batch files:

- **CSV**
  Character separated value ("CSV") format with the following formatting rules:
  - Text format files, using LF (line feed) or CRLF (carriage return and line feed) as record delimiters.
  - Field separators are either commas, semicolons, or tab control characters. Notice that IBM Safer Payments interprets the file separator used in the header line and utilizes this for the entire file. You may thus not mix delimiters.
  - The first row contains all mapping definition alias names of the message of the load job, separated by the field separator.
  - Notice that the data files must have the ".csv" suffix in their file names.
- **FCD**
  Fixed column data ("FCD") format using text format files with fixed length data ranges (no delimiters). Unlike CSV files, such data files do not contain a header row that identifies which variables are contained in each data row and what their sequence is. This information must be provided with format definitions from mappings. In this case, the start and length of each variable to be mapped to an IBM Safer Payments attribute are defined directly with the respective message mapping.
- **Json**
  Messages can also be imported in the JSON format. Every line must represent one complete JSON message, which can be parsed. "[" and "]" are forbidden characters for fieldnames, because these are used in the mapping to access array elements of the JSON

Value format

IBM Safer Payments supports different data types for which the following formatting rules apply:

- Numeric
  All numeric values may or not contain the period (".") as decimal separator. Negative values are preceded with the minus ("-"). No exponential format, no currency character(s), and no digit group separators may be present. The number of decimals used from the values delivered, and the minimum and maximum value depend on the settings of the respective attribute in the model. Values higher or lower than the limits are clipped to the respective limit.
- Text
  Must be put within quotes ("). All characters between the opening and closing quote are considered text value. The quote characters " may not be part of a batch data delivered value. Characters must also be in the ASCII range from 32 to 126 or 128 to 255. Values longer than defined with the respective model attribute are clipped to this length.
- Timestamp
  All timestamp values must be ISO-formatted, that is follow exactly the "YYYY-MM-DD hh:mm:ss" format ("YYYY": year in four digits, "MM": month in two digits, "DD": day in two digits, "hh": 24-hour in two digits, "mm": minutes in two digits, and "ss": seconds in two digits). If you need to enter a date as a timestamp, add " 12:00:00" to the ISO date format. Milliseconds are not supported.
- Time interval
  All time interval values use a format similar to ISO 8601 for each side of the interval and the "~" symbol to separate them. Several types of date and time information are supported but both sides of the interval must use the same type:
  - Full timestamps: Both sides are formatted exactly as you would when sending just a single timestamp. Example: "2018-03-01 09:00:00~2018-03-15 12:00:00".
  - Time only: Both sides consist of a time only. Possible formats are "hh:mm:ss", "h:mm:ss", "hh:mm" and "h:mm".
  - Day of the week only: Both sides use the english names of the weekdays either in full or abbreviated. Case does not matter. Example: "mon~wed".
  - Day of the week with time: Combines the type above with a time. All time formats mentioned above can be used. Example: "mon 9:00~wed 12:00:00".
  - Day of month and mont only: Both sides appear similar to full timestamps but with a "-" instead of the year and without a time. Example: "--03-01~--03-15".
  - Day of month and month with time: Combine the type above with a time. All time formats mentioned above can be used. Example: "--03-01 9:00~--03-15 12:00:00".
  - Day of month only: Both sides appear similar to full timestamps but with a "-" instead of the year and the month and without a time. Example: "----01~----15".
  - Day of month with time: Combines the type above with a time. All time formats mentioned above can be used. Example: "----01 9:00~----15 12:00:00".

If files are to be used as inputs for IBM Safer Payments that do not comply with these settings, they must be converted outside IBM Safer Payments.

Semaphore files

Data files used in the BDI typically are relatively large, so that it can take some time to write them in the incoming directory. If a load job would be started while the data is not fully transmitted, the load job could fail to load all records. Therefore the BDI supports "semaphore files". These files have the same name as the respective data file, but the suffix ".sem" . If the "Wait for semaphore file" checkbox of a respective job is checked, the loading of the data file will not started before the respective semaphore file is found in the incoming directory. Notice that the contents of the semaphore file is ignored; it thus is typically empty.

Interruptions and logging

During loading of a data file, a ".log" file is created that has the same name as the data file loaded. It contains a response entry for each record (line) of the data file that was successfully loaded. Each response entry is terminated by a line feed.

The response entry contains the output attributes and their values (if defined with the attribute mappings of the model revision), plus:

- the timestamp of the system time (UTC) the record was processed,
- the URID that IBM Safer Payments has associated the record with, and
- the computation (error) status.

If loading of a file ever gets interrupted during file loading, IBM Safer Payments uses the information of the log file to commence file loading where it was stopped.

Notice that processing of batch files can be interrupted (and later restarted) after each record. This allows shutting down an IBM Safer Payments instance before all jobs are completed and resume them at a later point in time.

However, if "Curtail masterdata" or "Re-create interval index" is enabled, the job will not be resumed after interruption. Instead, this will lead to a re-execution of the whole job. Interruption may not only be performed manually and when shutting down the instance but also during golives, saving of various elements (e.g. case classes, notifications, reminders...), updating of mandators, rebuilding of indexes, reloading of compliance lists. It is thus recommended to perform these kinds of jobs when there is low activity on the cluster.

Archiving and error handling

Once a data file is processed without errors, it is moved together with its ".log" and – if exists – its ".sem" file to the archive directory specified for this job.

If an error occurs, the respective file is moved to the error directory specified for this job. In this case, the records processed up to the moment the error manifests itself are logged in the ".log" file. If any of the archive or error directory paths is empty, the move is suppressed.

Messages

Records in a batch file are considered transaction messages just as those handled by the MCI. The "Message type ID" is the same (numeric) value used by IBM Safer Payments to identify the type of transaction message (messages are created and maintained in their own section on the administration tab, see messages for details). Message type ID can be associated with the BDI job either for the entire file, or read for each record (for more details, refer to jobs).

Importing encrypted job files

If "Job encryption enabled" is activated, encrypted job files can be imported through the BDI interface. See Importing encrypted job files for further details.

back to top

10.1.1.5 Status and Control Interface Overview

The SCI is used between IBM Safer Payments instances of a cluster to exchange status information and to dispatch control commands. It is hence the only IBM Safer Payments interface that cannot be deactivated.

During operation, each IBM Safer Payments instance can send "irisInstanceStatusRequest" type JSON formatted requests to all the other IBM Safer Payments instances to form a "complete picture" of the entire IBM Safer Payments cluster.

Direction

The SCI consists of both an incoming interface and an outgoing interface for each other IBM Safer Payments instance in the cluster. While you may enable/disable the incoming interface, the outgoing interface always remains enabled. If the target instance's incoming SCI is unreachable or disabled, the sender IBM Safer Payments instance considers this target instance as non-reachable (and displays this accordingly on the cluster page).

back to top

10.1.1.6 Encrypted Communication Interface Overview

The ECI is used between IBM Safer Payments instances of a cluster to exchange AES encryption keys. It is not needed for an unencrypted IBM Safer Payments installation. If deactivated, no exchange of encrypted keys is possible.

Direction

The ECI consists of both an incoming interface and an outgoing interface for each other IBM Safer Payments instance in the cluster. While you may enable/disable the incoming interface, the outgoing interface always remains enabled. If the target instance's incoming ECI is unreachable or disabled, the sender IBM Safer Payments instance considers this target instance as non-reachable (and displays this accordingly on the cluster page).

back to top

10.1.1.7 FastLink Interface Overview

The FLI (FastLink Interface) is used between IBM Safer Payments instances of a cluster to exchange configuration data and transaction data for keeping all IBM Safer Payments instances of a cluster at the same configuration and data level. It uses JSON (JavaScript Object Notation) formatted messages over IP networking. It transfers different types of messages:

- Cc-ed transactions
- Configuration and model revision updates

- System and audit event log messages

Buffering

Each IBM Safer Payments instance creates an outgoing queue that can temporarily store (buffer) the messages to be sent to the respective other IBM Safer Payments instance.

The queue buffer needs to be sized large enough that it can store messages for even multi-hour or multi-day outages or non-reachability of instances. During operations, each IBM Safer Payments instance writes all its messages that require replication to the other IBM Safer Payments instances (cf. list above) into this disk buffer. A separate service thread for each queue continuously reads from this buffer and transmits the messages to the target IBM Safer Payments instance. Once the target IBM Safer Payments instance has received and processed the message, it acknowledges this to the sending queue, which in turn erases the message from its buffer.

In normal operational conditions, there should thus only be a few messages "in transit". Thus a threshold can be specified with IBM Safer Payments settings which defines when IBM Safer Payments considers the number of messages in transit to be so low that it considers the IBM Safer Payments instances "in sync".

There are a number of situations in which messages can build up in the FLI outgoing queue buffers:

- Peak load
  Each IBM Safer Payments instance opens exactly one FLI outgoing queue connection for each other IBM Safer Payments instance. Thus all transmission of replication messages uses only one connection and is hence computed mostly in a single thread on the target IBM Safer Payments instance. In most IBM Safer Payments installations, however, there are multiple MCI connections and there may be multiple BDI jobs performing in parallel. It could thus be that the total throughout of messages processed by one IBM Safer Payments instance is (temporarily) higher than what one connection/servicethread to/at the target IBM Safer Payments instance can process. In this case, messages can build up in the buffer. As soon as the total throughput decreases, the buffered messages are transferred to the target faster than new ones come in and the buffer empties over time. Notice that any IBM Safer Payments installation must be sized so that the average transaction load does not exceed the FLI throughput as otherwise the buffers will overflow.
- Downtime
  Non-availability of an IBM Safer Payments instance in a cluster can stem from various causes, such as network outages, server/infrastructure hardware/software trouble, or from maintenance tasks. Certain maintenance tasks require an IBM Safer Payments instance to be taken offline or down:
  - Backup
    Certain backup strategies require the files of the IBM Safer Payments instance to be backed up to remain unchanged during backup, thus requiring temporary disablement of all incoming interface of this instance.
  - IBM Safer Payments live updates
    Certain IBM Safer Payments software updates can be made during full live operation of the IBM Safer Payments cluster. In this case, one instance by one is taken offline and shutdown, and restarted with the new software release.
  - Operating system updates/upgrades
  - Server hardware updates/upgrades

In all such cases, where one IBM Safer Payments instance is temporarily unavailable, the respective FLI outgoing queues buffer the replication messages so that once the IBM Safer Payments instance becomes available again, the FLI connections are automatically re-established and the buffer contents is transmitted. As soon as the number of messages in transit is below the defined threshold, the IBM Safer Payments instance is considered in sync. There are two timing parameters that can be set on the system configuration page to control the timing behaviour of this process.

Buffer sizing

The size of the FLI outgoing queue buffer can also be set on the system configuration page. It must be large enough to hold the replication messages for the maximum duration of an IBM Safer Payments instance non-availability you need to be covered for. If your configuration is using the deferred writing option, keep in mind that you will need the configured amount of memory both on disk and in main memory. Please consult with the IBM Safer Payments support if you are unsure about how to determine the correct size for your application.

Buffer management

If an IBM Safer Payments FLI outgoing queue buffer overflow occurs, e.g. if the outage of an IBM Safer Payments instance takes longer than sizing assumed, the IBM Safer Payments cluster has no means anymore to synchronize the non-available IBM Safer Payments instance. The outgoing queue buffers of the other IBM Safer Payments instances are thus dropped, and the non-available IBM Safer Payments instance must be re-created from the other instance; must be "recovered" as described as "cold start" procedure in cluster management.

To reduce the risk of a buffer overflow, IBM Safer Payments automatically takes certain measures when buffer space becomes scarce:

- BDI brake
  With the batch data interface section on the system configuration page, you may define a percentage threshold. If the buffer filling level of at least one FLI outgoing buffers passes this threshold, the batch data loading jobs on this instance will be frozen This ensures that the FLI buffers are not used up by data that is already stored on disk. All jobs continue loading once the threshold is no longer exceeded. Typical values for this threshold range between 10% and 50%.
- MCI connection/MQ closing
  With the FastLink interface section on the system configuration page, you may define another percentage threshold for MCI/MQ closings. This threshold is typically set between 75% to 95% and thus much higher than the BDI brake threshold and once the buffer filling level of at least one FLI outgoing buffers passes this threshold, message command interface connections are closed and MQ stops pulling messages. In a typical IBM Safer Payments cluster setup, this would cause the connected systems to route their transaction messages to the message command interface of another instance that may have its FLI buffers less filled. If this is not the case, the connected systems will not find an IBM Safer Payments instance in the cluster that will accept their transaction

messages and will react accordingly. In this situation, transaction messages may get lost. Yet it is important to "reserve" a small amount of buffer space for replication messages that deal with configuration changes. If such messages get lost because of a buffer overflow, the IBM Safer Payments instance concerned must be recovered in a complex process.

Notice that there are status alarm indicators that let you monitor the FLI buffers and that can alert administrators to potential buffer overflows before they occur.

Direction

The FLI consists of both an incoming interface and outgoing interfaces (queues) for each other IBM Safer Payments instance in the cluster. While you may enable/disable the incoming interface, the outgoing interface always remains enabled. If the target instance's incoming FLI is unreachable or disabled, the outgoing interfaces automatically buffer all data and resend it once the target's incoming FLI becomes responsive again.

Instance authentication via checksums

During startup cluster instances exchange an encryption key via the ECI. When sending FLI messages, a checksum for each message is computed, encrypted and also sent. The receiving instance uses the key to decrypt the checksum and validate it. The validation of checksums can be deactivated in the "System configuration". When enabled only authorized systems (i.e. the cluster instances) can send out FLI messages and change the cluster.

back to top

10.1.1.8 Relational Database Interface

The RDI is optional. During standard operation, IBM Safer Payments only uses its multi-tiered, cached data repositories (aka CDC, MDC, DDC) and thus is self-contained. From these repositories, data can conveniently be retrieved using IBM Safer Payments' query function.

However, IBM Safer Payments data storage has been optimized for IBM Safer Payments' real-time and analytical needs. Thus customers seeking the type of access to their IBM Safer Payments data that a relational database management system provides can connect any standard database management system (DBMS) to IBM Safer Payments.

To ensure that DBMS operations can never disturb IBM Safer Payments real-time operations, the RDI is designed as a file interface, where IBM Safer Payments generates SQL DML (data manipulation language) statements that can be read by a DBMS. The necessary DBMS loading scripts are not part of the IBM Safer Payments product.

An example of such a statement could look like (depending on the configuration) as follows:

```
INSERT INTO DEFAULT_TRANSACTIONS (`PRIMARY_URID`, `PRIMARY_INSTANCE_ID`, `SYSTEM_TIMESTAMP`,
`INPUT_ATTRIBUTE_1`, `INPUT_ATTRIBUTE_2`, …, `OUTPUT_ATTRIBUTE_1`, … ) VALUES (101, 1, '2016-09-01 09:45:00',
'Input_Value_1', Input_Value_2, …, 'Output_Value_1', …)
```

For table definition for cases see below

Configuration

- To ensure the correct syntax and escaping choose the database format from the list of currently supported formats.
- The encrypted values can be masked in the SQL DML statements by enabling the checkbox.
- The file creation interval can be selected for file per (second/minute/hour). Standard is one file per hour.
- If needed, a line break can be added after a defined number of characters. To enable this, enter the number of characters after which at least a line break should be set. If a text values length is greater than the set value, it will be truncated by defined line break length - 2, because the quotes need to be considered here. If it is set to zero, no line breaks will be added. There will always be a line break after each statement

Transaction data configuration

- The RDI can be enabled for transactions with the checkbox. There will only be SQL DML transactions statements created for mandators that have 'Transaction Settings' enabled.
- Only the selected attributes will be included in the SQL DML statements.
- The table name for transactions has to be the same name as the table in your database. The default name is "DEFAULT_TRANSACTIONS" but you can choose a different name.
- The SQL DML files for transaction data will be saved in the location of the delivery path.

Cases data configuration

- The RDI can be enabled for cases with the checkbox. There will only be SQL DML cases statements created for mandators that have 'Cases Settings' enabled. Some data will be exported into separate tables, like Audittrail, Reporting Attributes, Blocklist hits, hitting Urid, collusions fired, first parties and rules fired.
- The table name for cases has to be the same name as the table in your database. The default name is "DEFAULT_CASES" but you can choose a different name.
- The SQL DML files for case data will be saved in the location of the delivery path.

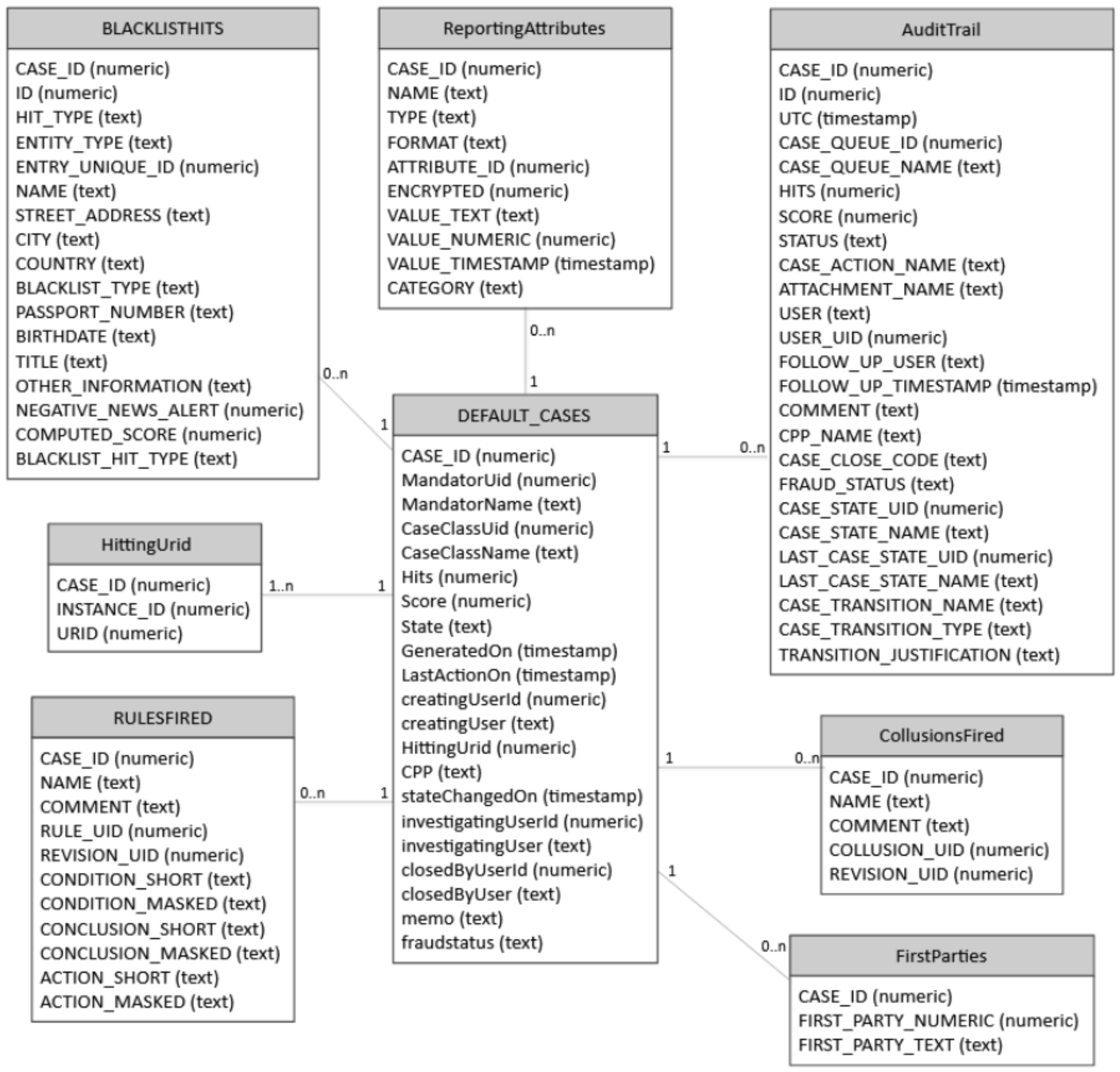
File name convention

All SQL files generated have the name "sp_transaction_data_*i*_*m*_*YYYY-MM-DD_hh-mm-ss*.sql" where *i* is the IBM Safer Payments instance id, *m* is the name of the mandator and *YYYY-MM-DD_hh-mm-ss* is the timestamp the file was generated.

Remarks

- Each INSERT and UPDATE statement is terminated by a line feed.
- Attribute names are converted to SQL standard (all caps, whitespaces replaced by underscore characters, leading numbers preceded by underscores, etc.). Best observe some IBM Safer Payments generated SQL statements to verify that the attribute names are what you expect them to be.
- All selected attributes are included in the INSERT and UPDATE statements if they are not null (value was set by either the delivered transaction message/record or computed by profiling or rule conclusion). Exceptions to this are the selected Boolean, IP and Hexadecimal attributes which will be included in any case. Null value attributes are not included so that the respective values in the database are also null.
- The "Primary Urid" and "Primary Instance Id" attributes are always delivered with each SQL DML statement, even though they are not explicitly defined attributes in IBM Safer Payments.
- Notice that the IBM Safer Payments generated SQL files require the database structure to be already established within the DBMS.
- The selected input/output attribute names of the champion are used as column names.
- A new file is generated according to the settings every second, minute or hour. The file name timestamp thus indicates that exact time. If there was no SQL DML statement for a certain second, the respective file is not generated.
- There is no removal function for outdated SQL files in IBM Safer Payments. You thus need to use for instance a script for feeding the files' contents into the DBMS and archive/delete them.

Table definition for cases



back to top

10.1.1.9 Alert Message Interface Overview

IBM Safer Payments has an outgoing message interface for alert messages. Alert messages include:

- Status alarm indicator (SAI) alert messages
- Investigation alert messages (Case actions, External Queries)
- Processng alert messages (Notifications)

Alert messages are sent by IBM Safer Payments using one of the following protocols:

- file system (plain text)
- file system (docx)
- HTTP message
- IP message
- ODBC SQL
- SMTP (email, SMS)

For each of these protocols IBM Safer Payments uses a dedicated outgoing queue, which sends the produced messages using asynchronous, parallel processing. The AMI is an outgoing interface. While it is possible to work with responses and response codes, the AMI does not provide any functionality to connect to Safer Payments from external applications, such as mail or database servers. To send information to IBM Safer Payments, use the BDI, MCI, or MQI.

Because it is assumed that all IBM Safer Payments instances use the same SMTP server, its configuration is made in the "settings.iris" file (administration system configuration page) rather than in the "cluster.iris" file (cluster administration page). This configuration will then also be offered as a template to all outgoing channel configurations using the SMTP protocol. Activation and de-activation of the AMI or specific protocols are controlled by the cluster administration page and stored individually for every instance.

back to top

10.1.1.10 WebSphere MQ Interface

The WebSphere MQ Interface ("MQI") connects IBM Safer Payments to the messaging solution WebSphere MQ.

WebSphere MQ enables IBM Safer Payments to receive transactions from a queue and write responses to another. The application writing the transactions and reading the response is independent of IBM Safer Payments.

Notice that further information on WebSphere MQ is provided in the appropriate WebSphere MQ documentation.

System Requirements

To use WebSphere MQ Interface, you need an existing MQ environment. This includes a MQ server installation in your network and a MQ client version 8.0.0.5 (or later) installed on your IBM Safer Payments machine. In order to enable IBM Safer Payments to make use of the client installation, "libmqic_r.so" has to be located on (or linked to) the library search part of your IBM Safer Payments installation. It is not possible to use WebSphere MQ Interface without an existing MQ installation.

Troubleshooting

The MQI will print out event log message 648 whenever it connects to a queue manager or a queue. Errors during message processing are reported with event log messages 184 and 441. If a connection is not possible, event log message 649 will be printed containing MQ reason codes. Please visit the IBM MQ Knowledge Base for more information about those codes.

back to top

10.1.2 SSL Settings

This section lets you define the SSL settings for this interface.

IBM Safer Payments needs two external contents to support an encrypted connection: certificate and private key files. In PEM format, both contents can be in the same file. If you are using encrypted certificates, IBM Safer Payments needs its passphrase to unlock the certificate. You may choose to either have IBM Safer Payments ask the operator for this passphrase each time it starts up, or a (secure) file location where the passphrase is stored.

Form Settings

- **Certificate file**
  The server certificate file in PEM format. This file has to fit with the Interface IP address, if the certificate validation is enabled
- **Certificate private key file**
  The private key file of server certificate in PEM format. This file can be the same file, as the server certificate file.
- **Diffie Hellman file**
  (Optional) The static diffie hellman file. This file is only needed, if you like to have a static diffie hellman key exchange.
- **Certificate passphrase entry**
  - *Passphrase input via console during startup:*
    You will have to insert the passphrase over console, every time IBM Safer Payments starts.
  - *Read passphrase from file during startup:*
    IBM Safer Payments will read the passphrase from an unencrypted file on startup.
  - *Use unencrypted private key:*
    The private key of the server certificate is unencrypted. No need to insert a passphrase.
- **Reject TLS 1.0**
  Reject encrypted connections using TLS 1.0. Reject TLS 1.0 is recommended as countermeasure against the known "BEAST" vulnerability in CBC ciphers.
- **Reject TLS 1.1**
  Reject encrypted connections using TLS 1.1.
- **Validate server certificate**

(Only in ECI) Validate the server certificate of outgoing connections against rfc 2818.

- **Server CA certificate file**
  (Only in ECI) The certificate of a remote server (in outgoing connections) must be signed by this Certificate Authority.
- **Server CRL file / path**
  (Only in ECI) The certificate of a remote server (in outgoing connections) must not be rejected by this Certificate Revokion List. This CRL is one file, that contains all revoked certificates.
- **Validate client certificate**
  Validate the client certificate of incoming connections.
- **Validate client certificate CN (API only)**
  Checks if the installed client certificate's Common Name (CN) fits with the users "login". The login is rejected with "login failed", if the CN does not match the user's login.
- **Client certificate file**
  (Only in ECI) The client certificate file in PEM format. This file is used for outgoing connections and will be verified by another IBM Safer Payments instance.
- **Client Certificate private key file**
  (Only in ECI) The private key file of the client certificate in PEM format. This file can be the same file, as the client certificate file.
- **Client Certificate passphrase entry**
  (Only in ECI)
  - *Passphrase input via console during startup:*
    You will have to insert the passphrase over console, every time IBM Safer Payments starts.
  - *Read passphrase from file during startup:*
    IBM Safer Payments will read the passphrase from an unencrypted file on startup.
  - *Use unencrypted private key:*
    The private key of the client certificate is unencrypted. No need to insert a passphrase.
- **Client CA certificate file**
  The certificate of an incoming connection must be signed by this Certificate Authority
- **Client CRL file / path**
  The certificate of an incoming connection must not be rejected by this Certificate Revokion List. This CRL is one file, that contains all revoked client certificates.

Remarks

- Since IBM Safer Payments often is operated as a Windows service or as a UNIX daemon, a console (window) for password entry is often not available to IBM Safer Payments. In this case, the password must be read from a file.
- If you store the passphrase in a separate file, this file must be protected from any access other than the IBM Safer Payments process.
- Notice that the SSL settings are individual for each IBM Safer Payments instance. This is also because different instances of IBM Safer Payments running on different computers with different IP addresses will require different certificates.
- Enabling/disabling of SSL (above) and saving your settings is immediately carried out.

Certificate validation

To avoid man-in-the-middle attacks, it is recommended to use certificate validation.

- **Server certificate validation**

  The server certificate validation is to validate the IBM Safer Payments server. Without this check, the client software cannot distinguish between a real IBM Safer Payments connection, and a man-in-the-middle connection.

  For the *API* and the *MCI*, the server certificate is verified by the client software (web-browser or message system). This means that you have to verify that the IBM Safer Payments certificate can be validated by your client software. This is usually a check against the configured CA of the client software and a check of the IP (or domain) against the CN field of the IBM Safer Payments server certificate
  For the *ECI*, the server certificate is checked by IBM Safer Payments against rfc 2818. This means, that the IP-address of the ECI and the CN field in the certificate of the IBM Safer Payments instance must be the same. Every ECI interface needs its own certificate with the fitting IP of the network interface. It is helpful, to name your certificate after the IP address to distinguish the different certificates of different instances.

- **Client certificate validation**

  The client certificate validation is to validate incoming connections.

  - *API*: The client certificate validation can act as a "two-factor-authentiation" token. Every user can get its own client certificate, which is checked against the CA and the CRL in IBM Safer Payments. All API connections without valid client certificate will be rejected in IBM Safer Payments, if the certificate is expired or not fitting the requirements.
  - *MCI*: The client certificate validation verifies, that no other system is sending transaction messages to the IBM Safer Payments MCI.
  - *ECI*: No other system should send "change encryption key commands" to IBM Safer Payments. The client certificate validation checks incoming ECI connections are sent from a valid IBM Safer Payments instance.

back to top

10.1.3 Storage Architecture

IBM Safer Payments uses a "three plus one" layer storage architecture to deliver exceptional real-time performance as well as ultra-fast simulation and rule generation results:

- A computational data cache (CDC) caches one complete transaction message or record during the entire computation. Once completed, the CDC content is copied to both the MDC and the DDC. Because the CDC uses the same binary data representation as the MDC/DDC, this copy operation is computationally very efficient. This enables effective parallelisation of all computation tasks, so that IBM Safer Payments fully exploits the power of today's multi-core computing hardware. The configuration of the CDC for MCI, BDI, and simulation/rule generation is made by respective settings in the IBM Safer Payments configuration.
- A memory data cache (MDC) stores data for two purposes:
  1. providing recent records for profiling (counter computation etc.) needed for real-time operations and
  2. providing data for simulation, analysis and model generation. Like the CDC, the MDC stores data in RAM.
- A disk data cache (DDC) stores data for two purposes:
  1. to provide a longer time period or more attributes than the MDC (mostly for case investigation) and
  2. to prime the MDC upon IBM Safer Payments startup.

  Because the DDC stores data in files, it is much slower than the MDC but it can be much larger.
- A relational database (RDB) can be optionally added with later releases of IBM Safer Payments. While it is not necessary for IBM Safer Payments operations, it can provide IBM Safer Payments users with the capability to run custom long-term analyses and reports without any disturbance to IBM Safer Payments operations. IBM Safer Payments only delivers data to this storage layer, even though data can be exported from the RDB and re-imported into IBM Safer Payments.

Notice that IBM Safer Payments calls its disk storage facility a "cache". This is because – like any cache – it only stores data for a specified amount of time. The reasons for this, and the techniques used, are described below.

Test data

In addition to the CDC/MDC/DDC layered cache structure, IBM Safer Payments also has a data cache for test data, called the SDC (Sandbox Data Cache). This is not linked to the CDC/MDC/DDC that deal with real (production) data, but linked to a challenger model revision where it serves the purpose of testing the model behavior with test data. SDC data is therefore stored with a model revision and since it has no connection with the other data caches, it is not explained further here.

Interaction of caches

While MDC and DDC are structured by records, the CDC is structured by computational threads. This is because the purpose of the CDC is to maximise utilisation of computer hardware with a large number of cores. Without CDC, each message computation would have to access attribute values in the MDC many hundred times during the computation of the message. This would limit the number of parallel computing threads. Using a CDC, each message computation thread is completely independent up to the moment when it dumps into the MDC. Since this is only one single time per message, the number of computational processes that can perform in parallel is thus orders of magnitude higher.

While the CDC stores only one record, but *n* times for *n* computational threads, the MDC and DDC store multiple records, but just once for all threads.

The cooperation between MDC and DDC can be configured through the deferred writing option.

Sizing example

For illustration, a possible sizing of an IBM Safer Payments installation is considered:

- The DDC is sized to 180 days of data. This is a typical maximum time period transaction data is required for case investigation purposes. The DDC also holds all attributes.
- The MDC is sized to 60 days of data and only the subset of DDC attributes. This is because it typically takes about 30 days to have the majority of fraudulent transactions flagged, and this information is essential for analysis and model generation.

Notice:

- MDC/DDC sizing can be different for every attribute.
- The DDC size must always be greater than or equal to the MDC size, because to ensure fast startup, the MDC is primed with data from the DDC. Because both data caches use the same binary data representation, this priming is very fast.
- Because the DDC is on disk, memory is usually available in abundance. However, access to data within the DDC is significantly slower.
- The kind of fast access provided by the MDC is typically necessary for the computation of counters or the merging of transactions. Because here, a sequence of up to many hundred previous transactions must be evaluated, disk access for each of these transactions would make this operation too slow. It can therefore be defined with the respective model revision profiling method whether or not evaluation should consider DDC transactions in case the MDC stored transactions are not exhaustive.
- If the optional RDB layer shall be employed, IBM Safer Payments creates SQL insert scripts that can be loaded into a database. Following database systems can be used to import the SQL commands: IBM DB2, Oracle, MS SQL, MySQL.

Unique record ID

Messages that enter IBM Safer Payments either via the MCI or the BDI are stored as records in the cache (if they are no merging or masterdata sources). At the moment IBM Safer Payments creates a new record for this message, it associates it with a so-called "unique record ID" (urid). This ID is a number that starts with 0 for the first record and is then incremented with each new record.

The urid is used everywhere in IBM Safer Payments where a record must be identified. Internally it is used to address records in MDC and DDC. The urid are also provided externally so that each record can be uniquely referenced.

Data cache sizing

IBM Safer Payments lets you exactly configure how it shall use its disk and main memory resources to cache data. Because of these many degrees of freedom, some planning ahead is required to obtain optimum balance between resource usage and computational performance. This section explains the background to IBM Safer Payments disk caching that must be considered when planning data cache sizing.

Of the many data cache layers of IBM Safer Payments, only the MDC and DDC layer require sizing by the user. The other caches are managed by IBM Safer Payments internally and their resource consumption is typically orders of magnitude below MDC and DDC.

MDC and DDC both hold individual transaction data. They can be configured for each attribute and each index.

Attributes

With attributes, configuration is relatively easy:

- Attributes that you do not want to store in the MDC or DDC do not require any sizing and do not consume any resources. Examples of such attributes are profiling output attributes that in many cases are not needed anymore after a transaction is processed, and input attributes that are only used during the processing of a transaction. You may later recreate such attributes for analytical purposes using IBM Safer Payments’ simulation capabilities.
- Attributes that you do not need in real-time processing, that is, attributes for which you do not plan to define counter or merging conditions/conclusions, typically do not need to be stored in the MDC at all. You may enable counter and merging conditions/conclusions to be defined for DDC stored records in the IBM Safer Payments settings. In this case, the full history of the DDC becomes available for real-time processing; however, computational performance can significantly suffer as a result. If you size MDC for these attributes, query and other access to this attributes will be faster and main memory consumption will increase. To determine the DDC size for such attributes, take the number of records your processing generates each day and multiply it by the number of days you would like IBM Safer Payments to be able to access this attribute. While IBM Safer Payments allows you to define the DDC sizes differently for each attribute, it is common practice to define the same DDC size for all attributes that are stored in the DDC. This is different to the MDC because typically disk storage is not as scarce as main memory.
- Attributes that you want to access in real-time processing typically get stored in the MDC and DDC. MDC storage always implies DDC storage because the MDC gets primed with data upon startup from the DDC. Typically, the MDC is sized smaller – as main memory is more costly than disk space – and reflects the time period that counters or mergings would need to access this attribute into the past. Because of the costliness of main memory, MDC sizes are not commonly the same for all attributes. MDC sizes are "sized to fit".

While IBM Safer Payments allows you to define different MDC and DDC sizes for each attribute, this can be confusing. Thus it could be preferred to decide:

1. for a time period that you want certain attributes to be available in counter or merging conditions/conclusions and rule generation; and
2. for a time period you want certain attributes to be available for case investigation, queries, analyses, and simulation.

Multiplying each time period with the number of records results in (1.) the unified size for the MDC and (2.) the unified size of the DDC. You then divide attributes in the same three categories as presented before – not stored / only DDC / MDC and DDC – and can you apply these settings. If later you find that for certain attributes you need longer or shorter historical evaluation, you can then modify the MDC size.

Indexes

Sizing indexes is different because they do not store historical records, but index value entries, such as card numbers, account numbers or merchant identifiers. While these types of indexes are mostly used to allow IBM Safer Payments to evaluate transaction sequences of the value entries, indexes can also be used to profile general entries, such as country codes, merchant categories or POS entry codes. In the latter case, indexes would not have sequences.

The index itself is sized to fit the total expected entry values that occur during the lifetime of the index. This value is the same for MDC and DDC, and each index by definition is stored in both MDC and DDC because IBM Safer Payments must always store the entire index both on MDC and DDC. Notice that unlike attributes, index entries cannot just leave the cache because they are eventually overwritten by new records. Unlike with attributes, the "age" of an index entry is not when it was entered into the ring buffer, but when it was last accessed. The latter, however, is in no relation to the position of the entry. Thus simple overwriting of data cannot work. Therefore each index entity is given a "lifetime" in days. Once an entry has not been accessed (there was no transaction on this entry) for longer than this time period, IBM Safer Payments can overwrite this entry. This ensures that indexes are not constantly growing but contain all data needed for operations.

Such an index can be used for calendar profiles, masterdata, and events.

If you also need to evaluate transaction sequences for the index entries, that is to evaluate counters and mergings along these index entries, you need to enable the sequence feature of an index. The sequence is similar to an attribute as it is also stored on a record basis and always points (for this index) to the record "before" the current one.

It is for this reason that sequences have MDC and DDC sizes like attributes. Typically you want the index to reach over the entire time period of the MDC and DDC, respectively. In this case, you would define the MDC to be the maximum of all attribute MDC sizes, and the DDC to be the maximum of all DDC sizes.

back to top

10.2 Python Code Execution

It is possible to use the Python interpreter within IBM Safer Payments. That allows you to create your custom scripts (aka "modules") and define calls to the functions of that scripts directly from IBM Safer Payment's decision model. Allowing to run Python code provides a lot of flexibility and freedom in defining additional transaction processing capabilities where the whole power of the Python language can be used in conjunction with other model element types. You have full control over defining arbitrary functions specifically tailored for your needs which includes defining formulas but is not limited to performing much more complex operations. Transaction and profiling attribute data can be easily made accessible to your Python scripts which is explained in more detail in the subsequent paragraphs.

In order to use Python within IBM Safer Payments you need to have one of the supported Python versions installed on the host machine. Supported Python versions are 2.6, 2.7 and 3.5. IBM Safer Payments will automatically detect and link to the highest supported Python version during start-up. Upon successful load there should be an indicator in the system internals page.

Python execution can be enabled or disabled on a per mandator level. By default this setting is disabled and you can leave it disabled if this functionality is not needed. When enabled, Python scripts created offline can be directly uploaded into IBM Safer Payments using the respective subsection in the mandator form, and calls to these functions can be defined in model revisions of that mandator. Additionally, you can make each Python script also available in sub-mandators, and thus in the model revisions of sub-mandators respectively. By default, the use of each Python module is restricted within the target mandator only. However, it can be easily extended to its sub-mandators if decided to do so by ticking the respective checkbox next to the uploaded Python module name.

Structuring your Python code

IBM Safer Payments loads all functions that are found in the uploaded Python modules and makes them available to be used in model revisions. You thus need to wrap your code in functions. Each module can contain one or more functions. There is no limitation on using any of Python packages available on the host machine in your scripts. However, these packages have to be available at the moment of uploading the module otherwise the module will not be loaded and its functionality will not be available. Below you can find an example of a script containing a simple Python function that calculates the length of a given string:

```
def calculateStringLength(inputString):
  """
  This function takes a string as an input and calculates its length.
  Returns the number of characters in the input string.
  """
  return len(inputString)
```

It is generally useful to comment your Python functions and by that provide cues to other users of the system on what is each Python function doing exactly. To achieve that, you can use the Python comment notation and add function description as shown in the example above. You might want to provide general information about what arguments does function take, how does it processes data and what is the return value. This comments are informational only and will be later shown the in the model revision where calls to Python function are to be defined.

Defining a call to Python function

Python functions can be used in rule conclusions and formulas. In order to define a call, you need to type the function name in the expression field. It is sufficient to type "py" in the expression field to get the list of all available Python functions. Function calls are defined in the following format:

```
py.{module_name}.{function_name}(arg1; arg2; ... ; argn)
```

Below you can find an example showing how the call definition for the aforementioned example will look like:

```
py.string_operations.calculateStringLength({Customer Name})
```

Similarly you can define Python function calls that take constant values as input along with attribute data. Below you can find an example call how to pass a constant string:

```
py.string_operations.calculateStringLength("John Jakob Jones")
```

Python formulas can also be used as arguments to mathematical operations as for example:

```
(10 - py.string_operations.calculateStringLength({Customer Name}))
```

The result of a Python call is written to either an expression attribute or formula output attribute, depending on where the function call was used. In general, any numeric overwritable IBM Safer Payments attribute can be used for that.

Uploading new modules and updating the existing ones

Each Python module should have a unique name in the mandator hierarchy. To upload a module you can click the upload button of the respective section in the mandator form or simply use the drag and drop function to upload one or multiple modules. The uploaded modules are directly loaded and made available for use.

Updating an existing Python module requires some additional steps. You need to do your modifications to the desired functions locally to the desired script. Then you can upload a new version of your script. Since module names should be unique, you can include version number in your filenames such as "string_operations_v1.py", "string_operations_v2.py", etc. Note that uploading a new version of your module will not update old functions. You need to manually update Python function usages in a challenger revision so that functions of the new module are called. After updating all usages in the challenger revision you need to promote it to be the new champion.

Data exchange and data type mapping

In order to pass data to a Python function, IBM Safer Payments will need to handle the data type conversion, i.e. all data types of passed arguments in a function call will be converted to their counterparts in Python. Data type conversion will be handled according to the following table.

SP data type	Python data type
Numeric (with decimals)	Float
Numeric (without decimals)	Long
Timestamp	Long
Text, IP*, Hexadecimal	String
Boolean	Bool

Similar to passing data to Python, the same rules of data type conversion apply when data is received from Python call-outs (return values). Each Python data type can only be stored in a matching IBM Safer Payments attribute as shown in the table above.

* While it is possible to pass IP attribute data to Python as "String" data type, it is not possible to use IP attributes for storing Python outputs of type "String".

back to top

10.3 Encryption and PCI DSS Compliant Operations

This section covers certain aspects on encryption, primarily with respect to PCI DSS compliant operations of IBM Safer Payments.

back to top

10.3.1 IBM Safer Payments Security

Software products in payment systems must be protected against security breaches that allow a non-authorized person to:

1. access payment and customer data
2. manipulate data, models, and audit trails
3. sabotage the function of IBM Safer Payments
4. use IBM Safer Payments to perform malicious actions to the user

While IBM Safer Payments has multiple interfaces that could potentially be abused for such security breaches, only the API is typically configured to be accessible outside the data center.

Because the API is connected to the user computer’s browser, it is a potential door to intrusion for manipulation. If IBM Safer Payments is deployed over the internet to users in other companies, the circle of potential contact points that could access the API can thus not be limited to trusted people. And even if the people can be trusted, their computers could be compromised.

Therefore IBM Safer Payments contains various measures of protection against such manipulation on different levels:

- **Application Server**
  A potential intrusion scenario is to utilize a weak point in the application server to inject malicious code. Because IBM Safer Payments does not use a standard application server, but a fully embedded, custom designed function, standard weaknesses do not exist.
- **Buffer Overflow**
  Another intrusion scenario is to exploit unprotected buffer resources in a software program. The application serving function in IBM Safer Payments uses fully protected buffers and thus does not expose such vulnerabilities.
- **SQL Injection**
  This technique uses flaws in a software program so that SQL instructions are passed directly and uncontrolled from the program’s application logic to the underlying SQL database. Because SQL is a powerful execution environment, such an attack could cause significant damage. IBM Safer Payments is fully protected against this as it does not use any SQL engine.
- **HTML Injection**
  This summarizes any technique in which character sequences are passed to the software program (typically as user entries) that when later displayed back to the user either execute some HTML command or Javascript commands. To protect from any such attack, the IBM Safer Payments server escapes HTML reserved characters in user inputs.
- **Cross-site request forgery (CSRF)**
  Cross-site request forgery, also known as a one-click attack or session riding, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts.] Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser. IBM Safer Payments supports session-specific cookie values that it checks with each request to protect against CSRF attacks.

Escaping

To ensure that no "unsafe" characters can be injected into its data elements, IBM Safer Payments employs a set of cascaded escaping mechanisms:

1. The IBM Safer Payments client (aka browser component) uses standard URL encoding to escape special characters like quote, curly/square bracket, or space with a percentage character followed by a two-character hexadecimal value of the single-character UTF-8 value. For instance, a quote is escaped to **%22** and a space to **%20**. Language specific characters are encoded to two such values; the letter Ü for instance in encoded to **%C3%9C**. Within values of text entries all quotes are escaped as **%5C%22**, which corresponds to **"**. This is necessary because quotes are the termination characters for text values in the JSON format used by the IBM Safer Payments client to send HTTP requests to the IBM Safer Payments server.
2. The IBM Safer Payments server decodes all these URL encodings with the exception of the quotes within text values. The **"** are also present when any IBM Safer Payments element is stored on disk as JSON because JSON uses quotes as text delimiters.

3. The IBM Safer Payments server responds to the IBM Safer Payments client using only JSON data streams. In these, the characters **& # < > ( )** are escaped as **&amp; &#35; &lt; &gt; &#40; &#41;** Quotes within text values remain escaped as **\"**.
4. These escape sequences are decoded within the IBM Safer Payments client for display and in text entry fields.

This escaping mechanism ensures that an attacker cannot introduce potentially malicious code from both an IBM Safer Payments text entry field or as HTTP API request.

back to top

10.3.2 PCI DSS Encryption

IBM Safer Payments supports PCI DSS compliant encryption for PAN input attributes.

Attribute encryption

To use any encryption feature, encryption must be enabled on the "system configuration" page. Once enabled, each attribute page displays a checkbox to enable or disable encryption for this specific attribute. Provided you have the respective privileges, you may turn encryption on and off.

If encryption is enabled for an attribute, only users that have the view clear values privileges may see the values unmasked. In masked values all digits except the first 6 and last four are replaced with an X.

Remarks:

- Encryption consumes considerable computational resources. You should thus keep encryption to only the attributes where you really need it.
- With text type attributes, only the first 16 characters are encrypted.
- Each storage of an encrypted attribute requires 16 Bytes minimum, regardless of the length of the attribute.

Keys and their subkeys

Encryption in IBM Safer Payments is facilitated by so-called "key triplets". Each key triplet contains the subkeys:

- Private key
- Left public key
- Right public key

To enable encryption, all three subkeys of a triplet must be present for the triplet to be activated and perform IBM Safer Payments encryption and decryption.

The private subkeys of triplets are stored in a file "key_*no*.iris" in the IBM Safer Payments "key" directory, while the public keys are kept secretly by human "keyholders". To ensure that no one person alone can activate a key triplet; the public key is divided into two keys for two keyholders.

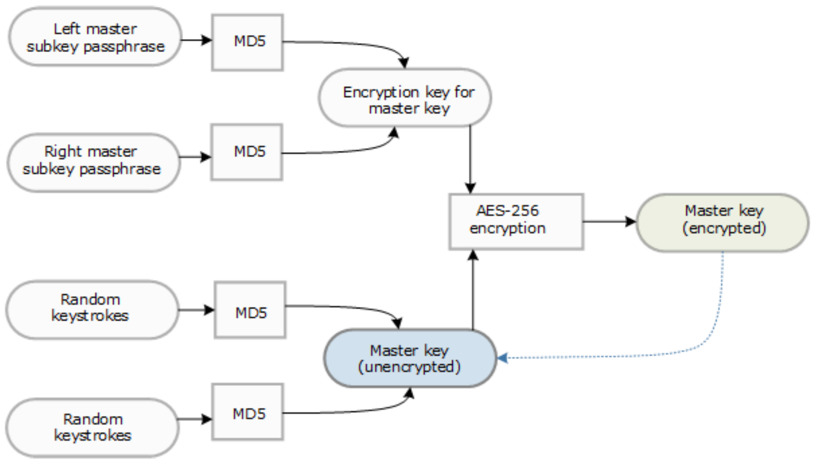
The naming convention "left" and "right" is to differentiate the subkeys and their "keyholders".

IBM Safer Payments allows keeping multiple active and non-active (a "non-active" triplet would be one where a subkey has not yet been provided) key triplets in its key management function, and allows for switching between the active ones. While only one of the key triplets can be active at a time, it makes no difference which of the key triplets is the active one.

Key triplets are differentiated by their number.

Master key generation

The following figure exemplifies the computational actions involved in master key generation:



The actual master key used by IBM Safer Payments to en-/decrypt data is generated by using two sets of random keystrokes hashed by MD5, delivering a 256 bit length root key. This master key is never stored or made accessible to users. Rather, using the two passphrases of the keyholders, the master key is encrypted using the AES-256 algorithm.

It is important to notice that (dotted line in the figure above) using the two passphrases, the encrypted master key can be decrypted.

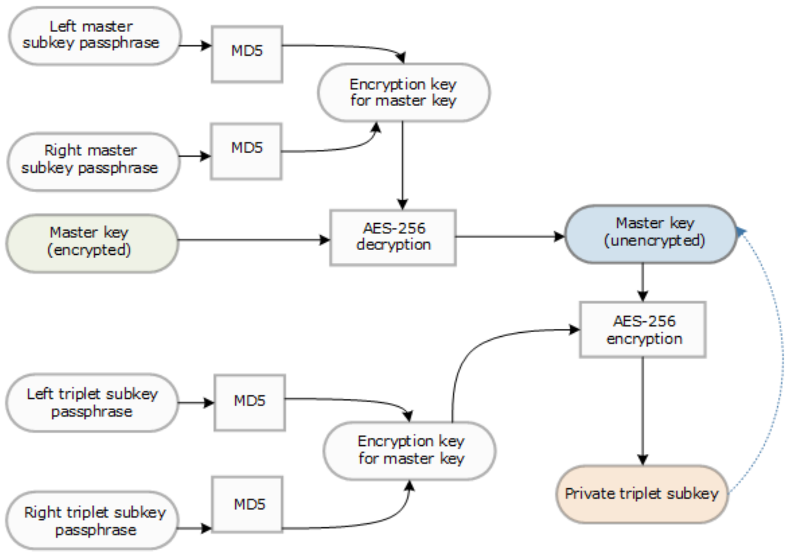
This encrypted master key is therefore stored in a safe place and is subsequently used – together with the passphrases of the keyholders – to create the triplets that are the only keys used during IBM Safer Payments operations.

This is also the reason why the key generator is provided as a separate utility program rather than an integral part of IBM Safer Payments. Not even the encrypted master key should ever be on the IBM Safer Payments server host. Rather you may use any other computer to create the encrypted master key, store it in a safe place, and generate triplets from this whenever needed.

Key triplet generation

Key triplet generation requires the left and right master key passphrases – thus the presence of the respective keyholders – plus two keyholders for the two public subkeys of each triplet. The keyholders may be the same people.

The figure below illustrates the process:


Left master subkey passphrase
MD5
Encryption key for master key
Right master subkey passphrase
MD5
Master key (encrypted)
AES-256 decryption
Master key (unencrypted)
AES-256 encryption
Left triplet subkey passphrase
MD5
Encryption key for master key
Right triplet subkey passphrase
MD5
Private triplet subkey

First the encrypted master key is read from file and using the two master passphrases gets decrypted in main memory only. From this unencrypted version of the master key now each triplet is derived by encrypting the master key with a new pair of passphrases.

The result of this process is the private triplet subkey. This must be stored in the "key" directory of the IBM Safer Payments installation. Because the file system of the IBM Safer Payments server host is a protected area, this provides an added level of security.

A good practice with key generation is to generate a number of key triplets in advance and then use them as they are needed.

It is important to notice that from each private triplet subkey, by decryption using the two public subkeys, IBM Safer Payments can reconstruct the master key in main memory for its operations.

Key management summary

IBM Safer Payments thus uses a 256 bit "master key" to encrypt and decrypt all data. Multiple key triplets are used to store an encrypted version of this master key. The master key is never stored by IBM Safer Payments on disk (it though exists in main memory during IBM Safer Payments operations in clear).

Key triplets always consists of a "private" key of 256 bit length, and two "public" keys of 128 bit length. All three complete keys are necessary to unlock the IBM Safer Payments encryption.

The private keys remain within IBM Safer Payments, the public keys are handed out to two different users.

IBM Safer Payments keeps a pool of private keys of active triplets. The private keys of active triplets are stored with the "key_*no*.iris" configuration files of IBM Safer Payments in clear (these files are not shared between the IBM Safer Payments instances automatically). As long as the private key of a triplet is in this pool, it is valid. That is, when the public keys are entered, the key becomes valid.

Key triplet generation is provided by a utility program delivered to IBM Safer Payments users. The first step is to generate a "master triplet". To generate the master triplet, the user enters two passphrases (or two different users enter passphrases) as the two public keys.

Notice that the use or passphrases for the public keys is strictly convenience. Passphrases can be memorized much easier than any binary key (passphrases can have any length; enforcement of "strong" passphrases is not part of the IBM Safer Payments software). The passphrases are hashed (MD5) into 128 bit public keys that are used by the utility to generate the private key stored with IBM Safer Payments.

Key administration

When IBM Safer Payments starts, its finds n active private keys in its "key_*no*.iris" files. To activate encryption, either at least one (fitting) public key pair must be entered, or IBM Safer Payments obtains the key from one of the other IBM Safer Payments instances in a redundant configuration.

Notice:

- They public keys are transported in clear between the IBM Safer Payments instances.
- The private keys are not transported between the IBM Safer Payments instances.
- New keys cannot be created from IBM Safer Payments administration.

Key management is controlled with 6 privileges that can be associated with a role:

- Key management
  View PCI DSS compliant encryption key management.
- Left key
  Entry of left public key.
- Right key
  Entry of right public key.
- Key activation
  Set the currently active key.
- Key revocation
  Invalidation of a key.
- Master key change
  Change a master key and re-encrypt all data.

Notice that no user may have the privilege to enter both keys.

For more details on role definition refer to the help page of roles.

Remarks:

- Each key triplet can be checked for whether the passphrases were entered correctly. The "key_*no*.iris" files therefore contain a known sequence so that correctness of passphrases can be verified before activation.
- Deactivated keys remain in the table "forever", only if they are revoked, they are physically removed. Deactivated keys can later be activated again.
- Only one key can be active at the same time, thus activating one key is deactivating all others.

Limited validity of keys

Both, key triplets and master keys are only valid for a limited amount of time. The key lifetime can be changed separately in the system configuration for master key and key triplets. IBM Safer Payments stores the time of the first activation of the key.

IBM Safer Payments closes all incoming data connections on all instances after key expiration.

The transaction processing or a subsequent setting change is no longer possible after expiration. It is strongly recommended to add a status alarm indicator which alarms the administrator before key expiration.

Cluster key handling

If encryption is enabled, starting a cluster is different:

- Once you start the first IBM Safer Payments instance, it loads its private keys. Because the public keys are not entered at this moment, this IBM Safer Payments instance does not set MCI, BDI, and FLI active (if IBM Safer Payments would receive transaction data through any of these interfaces, it could not store them encrypted).
- Therefore, you must first set a key active on the first IBM Safer Payments instance. This activates the enabled MCI, BDI, and FLI on the first IBM Safer Payments instance.
- Now, the other IBM Safer Payments instances can be started. Upon start, each IBM Safer Payments instance fetches all entered public keys with all other IBM Safer Payments instances via the ECI (encrypted communication interface). If an active key is found, this key is then also activated. This activates the enabled MCI, BDI, and FLI on the other IBM Safer Payments instances.

Once a cluster is up and running, and one instance is shut down, this instance upon re-start automatically "asks" the other IBM Safer Payments instances for the public keys. There should thus be no need to ever enter the public keys ever again as long as there is still at least one IBM Safer Payments instance running that stores the public key pairs.

Master key change

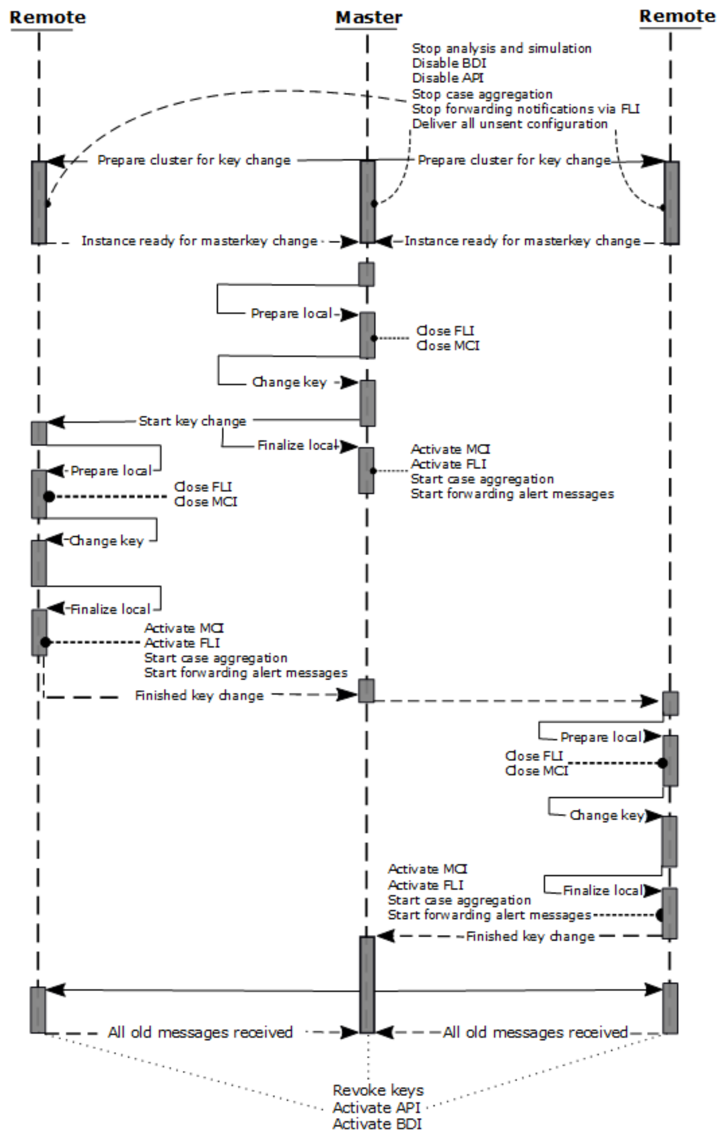
The master key change re-encrypts all encrypted data in IBM Safer Payments with a different key. To ensure that transaction processing is not interrupted during key change, IBM Safer Payments allows a cascaded key change. Similar to cascaded golives IBM Safer Payments delays the re-encryption on remote instances. The delay depends on the master key change wait factor which can be configured in system configuration. For a wait factor of "0" the key change is executed immediately on all instances. While this is the fastest way to change a master key it prevents IBM Safer Payments from scoring transactions during the key change. If the wait factor is greater than "0" the key change is executed on the instance with the active API at first. Once the first instance finished the remote instances are sorted depending on their FLI buffer fill levels and written to a list. The next key change will be executed on the remote instance with the lowest fill level. The other instances start the key change after a certain delay, which is defined as list position multiplied by master key change wait factor multiplied by the time it took to re-encrypt everything on the first instance. For a wait factor greater than "1" this leads to an cascaded master key change where only one instance re-encrypts at the same time. We recommend a wait factor of "1.05" to have an additional wait of 5% and to avoid having a simultaneous re-encryption on two instances.

If you want to execute a master key change follow those steps:

1. Generate a new master key with the keygen tool with a new master key ID and new private keys from this new master key.
2. Copy the new private keys ("key_<key_id_n>.iris") into the "key" folder on all instances
3. Do not replace any file of the "key" folder while copying.
4. Click on "reload private keys" in "Admin/Keys". You need the right to revoke keys to click on this button. The new master key should appear now.
5. Log out if necessary and let the left keyholder log in and insert the left key.
6. Log out if necessary and let the right keyholder log in and insert the right key.

7. Log in as your account. You need the right to activate a key.
8. Click on "change master key".

The figure below illustrates the process of a master key change:



Once the master key change is triggered all cluster instance enter a passive state. The information that the cluster is going to change the master key is send via SCI to ignore possible filled FLI buffers. Every instance performs the following steps to prepare itself for a key change:

- Disable the API and BDI.
- Stop the case consolidation. Alarms are still generated but will not be aggregated until the master key change is finished.
- Stop forwarding of notifications via FLI. Notifications are still created and will be send as long as the AMI is active on the generating instance. If there is no active AMI on the generation instance the notifications will be stored and forwarded once the key change is finished.
- Stop the reminder controller.
- Stop the end of day job controller.
- Write a "ready for key change message" to the FLI buffer that confirms the previous steps are completed.

The instance that had the active API waits in the passive state until it received the "ready for key change message" from all other instances and has sent its own. Afterwards the instance prepares itself for the re-encryption by disabling the MCI and the incoming FLI. In case deferred writing is enabled the deferred writing service is stopped as well and all files are dumped to disk. The re-encryption starts and the instance status is changed to master key change active.
During this re-encryption the instance dumps all configuration files, the encrypted ddc attributes, all active cases and all unsent notifications. Once the re-encryption was successful, the master notifies the remote instances to start the key change according to the previously described wait factor. The instance enters status finalize and reactivates itself again:

- Activate MCI and BDI. It is possible to process transactions.
- Activate incoming FLI. The outgoing FLI will send all messages that were encrypted with the "old" key. Messages that are encrypted with the new key are only send if the receiving instance finished re-encryption as well.
- Activate API and start reminder controller.
- Write all stored notifications and emails to FLI buffer and allow forwarding of alert messages.

- Start the end of day job controller.

Every instance reports when it received all messages that were encrypted with the previous master key. As soon as all instances have finished the key change and received all old messages, the instances revoke the previous keys and switch the status from finalize to ok. The master key change finished successfully.

Possible error scenarios
There are several error scenarios during a master key change:

- A remote instance is shutdown or crashes during the key change.
  If a remote instance becomes unreachable the master key change continues on the other instances, but the instances will wait until this instance is available again before leaving status finalize. Only instances in status invalidated are ignored. Once the instance is restarted it will start in status invalidated and allow the other instances to finish the key change. The restarted instance has to be restored.
- The master instance crashes during key change.
  It depends at which point the instance crashes. If the re-encryption is finished on the remote instances, the master instance can be restarted in status invalidated and the key change will finish successfully. If the instance crashes before the re-encryption was successful the remote instance will stay in status passive, but can be set to status ok using the maintenance function.
- The network communication to one instance is lost.
  IBM Safer Payments will wait indefinitely for instances that are unreachable. If necessary the other instances can be set to status ok via maintenance function to finish the master key change manually.

Maintenance functions
The master key change implements a special maintenance function to cancel a master key change on one instance. The function is bound to a special user privilege. It allows to cancel a master key change and set the new instance status to “invalid” or “ok” after cancellation.

Known error messages when reloading keys from disk

- "Active key cannot be found: it will not be accessible on next start"
  Current active key is not found on disk. Restore key files and reload keys from disk
- "The content of revoked_keys file is too short. Something went wrong while decrypting. Please contact the IBM Safer Payments support or change revoked_keys file!"
  Restore the revoked_keys file and reload keys from disk
- "The revoked_keys checksum is wrong. Something went wrong while decrypting. Please contact the IBM Safer Payments support or change revoked_keys file!"
  Restore the revoked_keys file and reload keys from disk
- "The "revoked_keys.iris" file cannot be found. Please contact the IBM Safer Payments support."
  Restore the revoked_keys file and reload keys from disk

back to top

10.4 Maintenance Functions

This section covers specific maintenance functions for IBM Safer Payments.

back to top

10.4.1 Rebuild Index

Re-creates an entire index based on transaction data stored.

This function can only be executed on an IBM Safer Payments instance that does not process any transaction messages or API requests (MCI, FLI, and BDI must be disabled on the instance). It performs the following actions:

1. Erases index in MDC and DDC.
2. The rebuild starts on the beginning (earliest record) in DDC and fills the index (and its sequences) from scratch (only in index MDC).
3. Dumps index MDC to index DDC at once.

Remarks

- Since with indexes (unlike attributes), MDC and DDC sizes are always the same. Thus the rebuild index function can create the "new" index only in MDC and then dump it at once to DDC; which is much faster compared to creating the index in parallel in MDC and DDC.
- The rebuilt index only contains entries for transaction records stored in DDC; all entries that are not stored in DDC will not be part of the rebuilt index.
- Depending on the size of the MDC/DDC, the execution of this function may take a significant amount of time during which this instance is not computing transaction messages.
- All index-oriented elements (calendar profiles, events) are recomputed as well.

back to top

10.4.2 Rewrite Element to Disk

Stores a serializable object of an IBM Safer Payments installation on disk.

back to top

10.4.3 Cleanout Revision

Unloads all non-champion revisions from IBM Safer Payments and moves their file representation from the "cfg" to the "arc" directories of all IBM Safer Payments instances.

This function is typically used in the implementation phase of an IBM Safer Payments installation, where when the staging moves from the test/QA/verification environment to the production environment to eliminate the potentially numerous iterations of revisions made during earlier stages.

Notice that if there is no champion revision for a mandator, one revision is not deleted to serve as copy source.

back to top

10.5 Miscellaneous

This section covers topics that did not fit into any other chapter.

back to top

10.5.1 Conditions

Condition format

Conditions in IBM Safer Payments follow the format:

attribute operator expression

where *attribute* is either an input attribute or an output attribute of one of the previous elements, *operator* is a comparator, and *expression* is either a single element, or a list of the following elements: A constant value (singleton), a wildcarded constant value, an interval, a mathematical formula or the reference to another attribute (special conditions like index based evaluation also allow masterdata, event and calendar computation as reference elements).

In a list of expressions you can combine all above elements. They will be computed with the operator OR. For example you can define a single condition that checks if an attribute value is within a list of constants, or equals any attribute of a list of attributes, or equals the result of any of multiple formulas.

Both the attribute and the operator can be selected via a drop down menu. The expressions can either be typed in manually or by using the context menu. To open the context menu, press the space key while the expression field is activated. All available types of expressions are listed in the context menu and can be selected directly. Furthermore, parts of the expression can be rearranged via drag-and-drop. This provides a quick and comfortable way to define conditions in IBM Safer Payments.

Operator overview

The table below shows a row for each operator. The columns explain which attribute data types for this operator are supported (empty field indicates that this combination is not supported; rest mouse over field to view details for certain operator/expressions).

Operator	Expression (single item or list of items)			
	Singleton	Interval	Attribute of same/current record	Math expression
[not] equal to	numeric / text / timestamp / hexadecimal / IPv4	numeric / text / timestamp	numeric / text / timestamp / boolean / hexadecimal	numeric / timestamp
greater than / less than [or equal to]	numeric / timestamp		numeric / timestamp	numeric / timestamp
[not] close to	numeric (±tolerance) / text (distance) / timestamp (±tolerance)	numeric ([x-tol; y+tol]) / timestamp ([x-tol; y+tol])	numeric (±tolerance) / text (distance) / timestamp (±tolerance)	numeric / timestamp
[not] starts /ends with	numeric / text	numeric / text	numeric / text	numeric / timestamp
[not] contains	numeric / text		numeric / text	numeric / timestamp
prefix [not] equal to	numeric / text	numeric / text	numeric / text	numeric / timestamp
[not] empty	This operator has no expression. Any value is considered 'empty' if it is not delivered.			
is true/is false	This operator is available for boolean attributes only and has no expression.			
same / distinct B / C net	These operators are available for IPv4 attributes only and expect an IP address or a list of IP addresses as expression.			

Remarks

- **Time interval** attributes can only be used on the right side of a condition and can only be compared to timestamps. All operators available to timestamps are supported.
- With the "[not] close to" operator, the "tolerance" setting is in percent for numeric type attributes. The "distance" setting for "text"

types is the minimum Levenshtein distance for the condition to be satisfied. For timestamp attributes the "tolerance" is a number without decimals. Its unit can be selected separately and is one of "seconds", "minutes", "hours" and "days".
- With the "prefix [not] equal to" operator, the number of characters considered to be the prefix characters is defined in a separate field of the condition row.
- IBM Safer Payments provides additional options for text type attributes. These are listed in a separate field of the condition row:
  - **Case sensitive/Ignore cases**
    Decides whether there is a difference between "TeSt" and "test" or not.
  - **Maximum/Minimum number of consecutive digits**
    Example: The maximum number of consecutive digits in "te123st45" is equal to 3 while the minimum number of consecutive digits is equal to 2.

Expressions using categories

If categories are defined for the attribute, the categories are accessible via the context menu. Each defined category leads to an entry in the context menu. Each entry is marked with an icon and has the format "*Label orignalValue*".

To use a category in a condition, you can either select the category in the context menu or type the original value directly in the expression field. In both cases the original value will be written in the expression field. To display the corresponding label, move the mouse over the value in the expression field. Notice that it is not possible to type in the label in the expression field. This will be treated as a constant.

Timestamp expressions

Timestamp attribute conditions have a few more options to define expressions:

- **Daytime expressions**
  Examples: "22:00~6:00", "22:00 ~ 06:00" or "06:00~22:00". Daytimes must always be intervals and they must use the 24h scheme (no "am/pm"). The first hour can be a single digit for times earlier than 10:00. Daytimes may either be expressed with or without seconds. If expressed with seconds, the format is hh:mm:ss. If the seconds are not expressed, they are set to "00" for-from daytimes and to "59" for to-daytimes. The interval "9:00~9:00" thus is equivalent to "9:00:00~9:00:59". Notice that minutes are not optional and the colon is the only supported delimiter.
- **Weekday expressions**
  Examples: "Monday~Friday", "Friday", or "sat ~sun". For the weekday, you may either use the unabridged "Monday" .. "Sunday" (first letter capitalized or not), or the three-letter abbreviation "mon" .. "sun" (first letter capitalized or not). It is important to notice that the begin of the from-weekday is considered 00:00:00 and the end of the to-weekday 23:59:59. Therefore, if you define "sun~sun" (or just "sun"), it translates to the entire Sunday.
- **Weektime expressions**
  Examples: "Monday 16:14~Friday 07:21". Weekday must be separated from daytime by a space character. The first hour can be a single digit for times earlier than 10:00.

There may be a list of these values (value pairs) in timestamp attribute conditions. The list may also combine singletons and intervals (e.g. "2009-12-22 05:22:53; sat~sun; Wednesday 14:09 ~ Wednesday 16:56; 00:10~00:20; 2010-01-02 00:00:00 ~ 2010-01-02 00:03:00" would be a valid timestamp attribute expression).

Wildcard expressions

The "[not] equal to" operator allows the use of wildcard characters for text and numeric type attributes. Possible wildcard constructs are:

- **Starts with**
  If the condition was "*attribute* equal_to text*", all attribute values that starts with "text" hit the condition. Note that this is equal to the condition "*attribute* starts_with text".
- **Ends with**
  If the condition was "*attribute* equal_to *text", all attribute values that ends with "text" hit the condition. Note that this is equal to the condition "*attribute* ends_with text".
- **Contains**
  If the condition was "*attribute* equal_to *text*", all attribute values that contain the substring "text" hit the condition. Note that this is equal to the condition "*attribute* contains text".
- **Bounded by**
  If the condition was "*attribute* equal_to te*xt", all attribute values that starts with "te" and ends with "xt" hit the condition.
- **Single wildcard characters**
  If the condition was "*attribute* equal_to A???in", each value that has the non-? Characters the same as the expression hits the condition (for instance "Austin"). Shorter words would not hit the condition.

The "[not] equal to" and "same / distinct B / C net" operators allow the use of wildcard characters for IPv4 type attributes. A single "*" can be used instead of a numeric block to include all 256 possible values for this block.

Remarks

- This implies that constants in combination with the "[not] equal to" operator may not contain characters "* ? ; ~", as they are used to mark wildcards, list elements, and intervals.
- In combination with other operators, wildcard characters ("*,?") are considered as "normal" characters in a text type constant.
- Double and single quotes are considered as "normal" characters in a text type constant. They will not be removed. White spaces are ignored.

References to another attribute

There are two different methods which reference to an attribute:

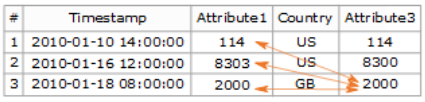
- {*attribute name*} - attribute of the current transaction
- [*attribute name*] - attribute of the same transaction

In situations in which only the current transaction is evaluated (example: rules) the two methods are equivalent. However, there is a difference whenever a sequence of transactions is evaluated (examples: counters, precedents, collusions, etc). The following example demonstrates the functionality of these methods:

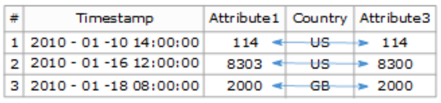
Assuming the following sequence of transactions:

#	Timestamp	Attribute1	Country	Attribute3
1	2010-01-10 14:00:00	114	US	114
2	2010-01-16 12:00:00	8303	US	8300
3	2010-01-18 08:00:00	2000	GB	2000

Here, the last row #3 represents the current transaction. The left table below (red) illustrates which values are compared when the condition *Attribute1 equal to {Attribute3}* is evaluated for the sequence. The right table (blue) shows which values are compared when the condition *Attribute1 equal to [Attribute3]* is evaluated for the sequence.



#	Timestamp	Attribute1	Country	Attribute3
1	2010-01-10 14:00:00	114	US	114
2	2010-01-16 12:00:00	8303	US	8300
3	2010-01-18 08:00:00	2000	GB	2000



#	Timestamp	Attribute1	Country	Attribute3
1	2010 - 01 -10 14:00:00	114	US	114
2	2010 - 01 -16 12:00:00	8303	US	8300
3	2010 - 01 -18 08:00:00	2000	GB	2000

Math expressions

All basic mathematical operations can be used in IBM Safer Payments in an intuitive way. Please notice that mathematical expressions must be surrounded by round brackets.

- (*Exp1* + *Exp2*)
- (*Exp1* - *Exp2*)
- (*Exp1* * *Exp2*)
- (*Exp1* / *Exp2*)

The operands *Exp1* and *Exp2* may either be:

- a numerical constant (may have leading minus sign)
- an absolute timestamp in the format YYYY-MM-DD hh:mm:ss
- an attribute of the current transaction (attribute name must be surrounded by curly brackets {*attribute name*})
- an attribute of the same transaction (attribute name must be surrounded by squared brackets [*attribute name*])
- a math expression itself

Formulas can be applied to numeric and timestamp type attributes. If values are used in conjunction with timestamp type attributes, the unit of expression values is seconds. Because when large time intervals shall be represented, this can be cumbersome, the letters "m", "h", "d" and "w" can be added to the value, indicating "minutes", "hours", "days" and "weeks", respectively. For instance:

```
TrxTimeStamp  less_than  (51w + {EmbossingDate})
```

hits when the current transaction is within less than 51 weeks after the embossing date of the card.

Remarks

Notice that each mathematical expression must contain exactly two operands. That means that *(a + b +c)* must be defined as *(a + (b + c))*.

Geographical Distances

In addition to the basic computation methods mentioned above, IBM Safer Payments is able to calculate the geographical distance between to geographical points. The format for this computation method is

- geoDistanceKm(pos(*latitudeA*;*longitudeA*);pos(*latitudeB*;*longitudeB*)) and
- geoDistanceMiles(pos(*latitudeA*;*longitudeA*);pos(*latitudeB*;*longitudeB*))

The operands *latitudeA*, *longitudeA*, *latitudeB*, and *longitudeB* are GPS coordinates in a floating point format (WGS84).

Example: *geoDistanceKm(pos(55.7522;37.6156);pos(48.8667;2.3333))* computes the distance (in km) between Moscow and Paris.

back to top

10.5.2 Conclusions

Conclusion format

Conclusions in IBM Safer Payments follow the format:

```
attribute  operator  expression
```

where *attribute* is any model attribute, *operator* is an assignment, and *expression* is either a constant value or the reference to another attribute.

Operators

If the element's conclusion is applied (all its conditions are evaluated to being true) with operator:

- **is**
  the value of *expression* is applied to *attribute* (all attribute types).
- **is (if not empty)**
  the value of *expression* is applied to *attribute* (not for bool, hexadecimal and IP). If the value of the expression is empty, the value of the attribute will not be changed.
- **increment by**
  the value of *attribute* is incremented by the value of *expression* (numeric and timestamp type attributes).
- **decrement by**
  the value of *attribute* is decremented by the value of *expression* (numeric and timestamp type attributes).
- **append**
  the value of the *attribute* (derived by computation of the transaction message so far) is appended to the value of *expression* (text type attributes). Notice that you may use this also to add delimiters to a list of items created by multiple applications of "append". Simply add another "append" conclusion with the delimiter as constant below the one appending the list item.
- **reset**
  the value of the *attribute* is reset to "nil", which computationally is interpreted as "0" for numeric type attributes, and as an empty string for text type attributes (all attribute types).
- **maximize with**
  the value of *attribute* is set to the maximum of value of *attribute* and value of *expression* (numeric attributes). This is usually used for scores. For example: The conclusion is *"Score maximize with 60"*. For a Score of 50 before the element's execution this would mean Score is set to 60 and for a Score of 70 this would mean nothing changes (i.e. Score remains 70).

Attribute values as expression

A conclusion may also copy the value of an attribute of the source transaction message to the target transaction record. For this, the source transaction attribute name is put in curly brackets into the expression field.

For instance, to transfer the value of an attribute "FraudReasonCode" from the merging source transaction message to the same attribute of the merging target transaction record, enter:

```
FraudReasonCode  is  {FraudReasonCode}
```

In merging conclusions it is also possible to copy the value of an attribute of the merging target transaction message to another attribute of the merging target transaction record, if the attribute in the expression is put in square brackets.

You may also use this to transfer (or increment, decrement, and append) the value of a different attribute of the source transaction message to the target transaction record. Notice that in this case, the attribute types must be identical. Different length or decimals are corrected by IBM Safer Payments automatically.

Formulas and Python function calls

You can also use all basic mathematical operations, geographical operations and Python functions in rule conclusions. To get the list of available Python functions you need to type "py" in the expression field. For more information refer to the online help.

back to top

10.5.3 Case Variable Conditions

Below you can find all case variables that can be used to define case conditions.

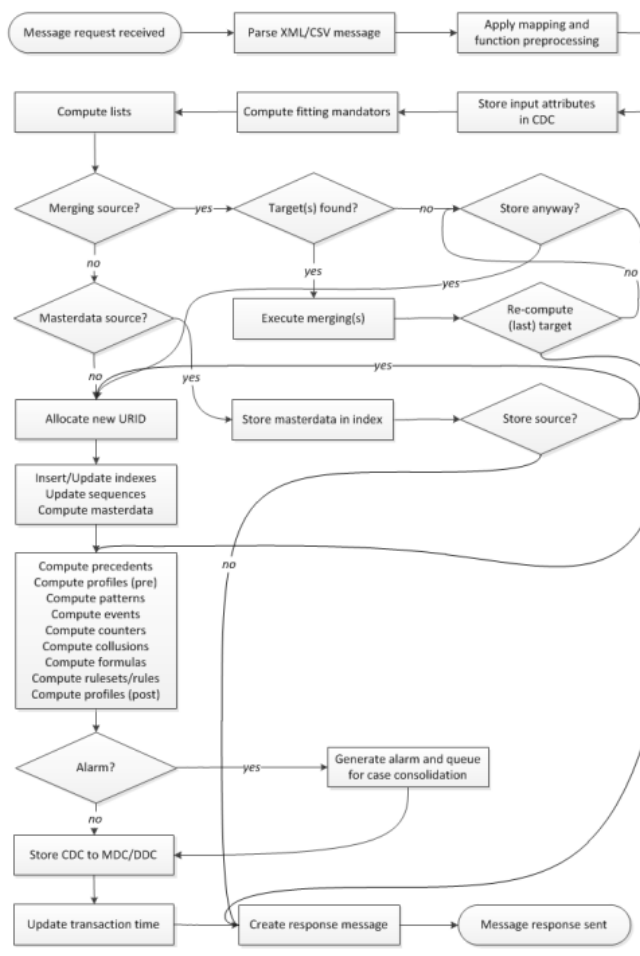
- Important dates:
  - [GeneratedOn]: The generation date as ISO formatted date.
  - [GeneratedOnTimestamp]: The generation date as UNIX timestamp.
  - [ClosedOn]: The case close date as ISO formatted date.
  - [ClosedOnTimestamp]: The case close date as UNIX timestamp.
  - [FollowupOn]: The followup date as ISO formatted date.
  - [FollowupOnTimestamp]: The followup date as UNIX timestamp.
  - [LastActionOn]: The last action date as ISO formatted date.
  - [LastActionOnTimestamp]: The last action date as UNIX timestamp.
  - [StateChangedOn]: The case state change date as ISO formatted date.
  - [StateChangedOnTimestamp]: The case state change date as UNIX timestamp.
- Timing metrics:
  - [CaseAgeInDays]: The time since case generation in days.
  - [CaseAgeInHours]: The time since case generation in hours.
  - [CaseAgeInMinutes]: The time since case generation in minutes.
  - [DaysSinceLastAction]: The time since last action in days.

    - [HoursSinceLastAction]: The time since last action in hours.
    - [MinutesSinceLastAction]: The time since last action in minutes.
    - [DaysSinceStateChanged]: The time since case state changed in days.
    - [HoursSinceStateChanged]: The time since case state changed in hours.
    - [MinutesSinceStateChanged]: The time since case state changed in minutes.
- User information:
    - [Investigating..]: The user, that is currently investigating the case.
    - [Viewing..]: The user, that is viewing the case and sending the case action.
    - [Closedby..]: The user, that closed the case.
    - [..UserName]: The username as string.
    - [..UserNameAndLogin]: The username, followed by the user login in parenthesis.
    - [..UserUid]: The system internal user UID.
    - [..UserEmail]: The users e-mail address.
    - [..UserPhone]: The users phone number.
    - [..UserLocation]: The users location.
    - [..UserMandator]: The users mandator name.
    - [..UserMandatorUid]: The UID of the users mandator.
- Other variables:
    - [CaseClass]: The name of the case class.
    - [CaseClassUid]: The UID of the case class.
    - [CaseClassId]: The ID of the case class.
    - [Mandator]: The case mandators name.
    - [MandatorUid]: The case mandators UID.
    - [Score]: The case score.
    - [Hits]: The case hits.
    - [FraudStatus]: The fraud status of the case close code, if the case was closed.
    - [CaseCloseCode]: The case close code, if defined.
    - [CaseCloseCodeUid]: The UID of the case close code.
    - [State]: The investigation state.
    - [StateUid]: The UID of the investigation state.
    - [LastState]: The last investigation state.
    - [LastStateUid]: The UID of the last investigation state.
    - [CaseUid]: The case UID, as visible in the case selection table (1-123).
    - [CaseUidRaw]: The case UID, as visible in url or in file system (1000000000000123).
    - [Memo]: The text value of memo field.

back to top

10.5.4 Message Computation

Computing responses to message requests is the essential function of IBM Safer Payments message request processing. The flow chart below illustrates this non-trivial process:



The steps in this computation process are:

1. IBM Safer Payments receives transaction messages either from XML requests via its IP port (real-time requests), WebSphere MQ via message queues or from data files (batch requests). With real-time requests, the message type Id is derived from the respective <IRIS> element attribute "MessageTypeId"; with batch requests, the message type Id is taken from the job configuration.
2. From the message type Id, IBM Safer Payments concludes the message definition to be applied. The message definition contains the alias names that IBM Safer Payments should look for in the transaction message received, and defines the function preprocessing to be applied.
3. Now all input attribute values are stored in the computational data cache (CDC) of the computational thread that performs the computation of this transaction message. The CDC contains storage positions also for the model output attributes and the output attributes of the profiling elements.
4. Next IBM Safer Payments assembles a list of mandators that shall be applied to compute this transaction message. For this, IBM Safer Payments first identifies the mandator this transaction belongs to, and then creates a list of all mandators on the path to the top mandator from there. This list is then computed top-down, that is starting with the top mandator to the transaction owning mandator. This computational sequence with respect to mandators is applied to all subsequent model element computation steps (lists, indexes, masterdata, calendar profiles, events, counters, formulas, and rulesets/rules).
5. Now IBM Safer Payments computes model lists.
6. Next IBM Safer Payments checks if the transaction message satisfies any of the merging source conditions of any applicable mandator. If so, it checks if any of the applying mergings has "insert" merging source activated. If so, an URID is allocated for the transaction message, the merging(s) is/are executed, the merging source transaction message as stored in the CDC is stored (permanently) into the MDC/DDC, a response message is generated, and the response message is sent. If none of the applying mergings has "insert" merging source activated, the merging(s) is/are executed, and a response message is generated and sent. This concludes the processing of a merging source transaction message.
7. Otherwise, the transaction message is checked if it satisfies any of the masterdata conditions. This process is analogous to the merging process.
8. If the transaction message is no merging and no masterdata source, it is a "normal" transaction and thus an URID is allocated for it. Then all model elements are computed, each for all applicable mandators. The generated output attribute values of all elements are stored in the CDC.
9. Once all computation is completed, IBM Safer Payments checks of the "CaseClass" meta attribute is set (nonzero). If so, an alarm is generated and this alarm is queued for case consolidation. Case consolidation is performed asynchronously by a separate computational thread dedicated to just case consolidation. Only after case consolidation, the case becomes "visible" in case investigation.
10. Finally the CDC gets "dumped" to the MDC/DDC, which renders the transaction message permanent as record. If the "Timestamp" meta attribute value of the transaction is later/higher than the one of the previous transaction, the IBM Safer Payments internal "transaction time" is updated. This time is used for period computation. Then the response message is generated with all status and output attributes of all applicable mandator models, and sent back to the service consumer (if it was a real-time transaction message) or written to the ".log" file (if it was a batch transaction message).

back to top

10.5.5 Time Representation

IBM Safer Payments is designed to work in a world environment, where transactions can come from any region (and thus time zone) as well as users. Therefore IBM Safer Payments uses three time representations:

- **UTC (universal time coordinated)**
  IBM Safer Payments uses UTC timestamps internally for all its representation of date/time data. UTC is also used between the IBM Safer Payments server and the IBM Safer Payments client (via the API), where the IBM Safer Payments client recomputes the local date/time for display.
- **Local time**
  Local time is used in any display and entry of date/time information by the user according to his preference settings. Therefore users in different time zones will see different date/time values.
- **Server time**
  Server time is used for display of any server relevant date/time information and as a reference for the actual transaction message computation.

back to top

10.5.6 Benchmarking Prevention Performance

To measure the performance of a fraud prevention system, two performance indicators are most relevant:

- Fraud detected
- False alarms

A fraud prevention system is better, if it detects more fraud and generates fewer false alarms. Unfortunately, both objectives are mutually exclusive in real world fraud prevention. Tuning a fraud prevention system to catch more fraud will also increase the number of false alarms and tuning it to decrease the number of false alarms will decrease the amount of fraud detected.

Therefore tuning a fraud prevention system is always about finding the best compromise between fraud detection rate and false alarms. The analytical features of IBM Safer Payments assist you in this task.

Frequently your tuning task will be determined by external constraints. For example, you may be in a situation where you are only allowed to a certain alarm rate.

Multiple definitions of the benchmarks listed above exist. The remainder of this document uses the following definitions:

Fraud detected

Fraud detected is measured either for individual rules, rulesets or analysis categories as a percentage of the total fraud amount. If there is $1000 of fraud in a given set of data, and a rule would hit $10 of it, the fraud detection would be 1%.

Therefore if you get the result that a given ruleset will deliver you 50% fraud detection, you would have saved half of your fraud losses if this logic would have been used.

False Alarms

False alarms are measured by the ratio of false alarms that are generated for any correct alarm (hit). If the false alarm ratio is 5, there would be 5 false alarms for any hit.

Notice that in the example above, a total of 6 alarms is generated.

Also notice that fraud detected is using "amount" as reference, while false alarms uses "number of transactions" as reference. This is because for fraud detected shall reflect the monetary savings while false alarms shall reflect customer disturbance. While monetary savings are expressed by the loss amount saved, customer disturbance is expressed by the number of genuine transactions that are intercepted.

Savings per false alarm

"Fraud detected" is measured either for individual rules, rulesets or analysis categories as a percentage of the total fraud amount. If there is $1000 of fraud in a given set of data, and a rule would hit $10 of it, the fraud detection would be 1%.

However, "fraud detected" cannot be the only performance measure of rules and rulesets. A rule may "catch" a lot of fraud, but if it at the same time generates too many false alarms, it is not a "good" rule. Therefore it has become customary to use a second benchmark figure, dubbed "false positive". False positives are typically measured as the number of false alarms that are generated to generate one correct alarm.

Notice that while "fraud detected" uses the amount of the transactions as measure unit, the "false positive" use the number oftransactions.

Tuning a decision logic always takes these two benchmark figures into account, and you will find them on many pages in IBM Safer Payments. Always one implies the other. In theory, you can catch all fraud if you intercept with every transaction. This would at the same time result the worst result for false positives. No false positives are generated if you never intercept with any transaction at all. It is obvious that the setting you want your decision logic to have should be somewhere between these extremes. The question is, where.

Often you have constraints with one of the two benchmark figures. You may be in a situation where you are only allowed to intercept with one in a thousand transactions. With your false positives limited to this figure, you now try to get the best fraud detected rate possible for the interceptions permitted. Or you have a fraud savings target, in which case you try to achieve this target with the minimum number of (false positive) interceptions.

Whatever situation you are in, however, a good rule is always one with a high "fraud detected" rate and low "false positives". In this situation it is beneficial to use a single performance indicator to express how "good" a rule really is. IBM Safer Payments introduces a new performance indicator to rules, rulesets and decision logics, the so-called "Saved amount per false alarm". This performance indicator is a monetary figure. If the value for a rule or a decision logic for this benchmark is "$10", it means that for each false alarm, you save $10. This performance indicator does not say whether this rule only saves $10 and generates one single false alarm or whether it saves $1,000,000 and generates 100,000 false alarms. It is a "relative" benchmark.

What this performance indicator is good for, is to sort rules with respect to their quality. IBM Safer Payments' analytical capabilities let you generate a list of all your rules, sorted by the "Saved amount per false alarm" (SAPFA). If you now select rules starting with the highest value, you are always sure that you prefer the rules that generate most fraud savings for the false alarms ("most bang for the buck").

back to top

10.5.7 Online Help

IBM Safer Payments offers three ways to access online help:

1. Context help
   Context sensitive help is directly available from the toolbar of the most page section by clicking on the respective icon.
2. Topic help
   Generic topic help is available either from context help pages or from the links below.
3. Search help
   If you need help to any subject, just type in the keyword(s) in the light blue box in the upper right hand corner of this (and any) help window. As soon as you type in three letters or more, a drop list of suggested choices appears. Pressing the return key opens a full page of search results.

Usage hints

- All online help pages appear within the browser window you opened for IBM Safer Payments. To quickly identify help pages, their header uses bright green background colour.
- They always appear over the IBM Safer Payments page and can be moved by dragging the header bar with the mouse.
- To change the size of a help page, use the drag bars at the right side and bottom of the help page. Help pages scroll only vertically.
- Notice that each help page's position and size settings are stored individually in your user account preferences. Thus, next time you open the same help page, it will open at the same position and with the same size.
- A new browser window with a printer optimized formatting opens by clicking on the toolbutton at the right side of the help page header. Use the print function of your browser to print the page.
- Help pages are closed by clicking on the toolbutton at the right side of the help page header, or by pressing the [Esc] key.
- Links to other help pages are shown in blue; underlining appears when the mouse pointer rests over them. If you click on a link, the new help page is displayed on top of the others. IBM Safer Payments keeps all online help pages open until you manually close them, even when you navigate to another IBM Safer Payments page. Notice that there is no "back" navigation in help pages as the browser "back" function is tied to the IBM Safer Payments page itself.

IBM Safer Payments manual

You may generate a printable version of all online help pages, formatted and structured as a manual by clicking on the toolbutton at the right side of this help page header. The IBM Safer Payments manual opens in a new browser window. You may either print it from there or import it to a word processing software for further processing. For example, to import the manual into Microsoft Word, click into the manual browser window, press [Ctrl]-[A] to mark the entire text, press [Ctrl]-[C] to copy it to the clipboard, and press [Ctrl]-[V] in a new empty Word document. Notice that Word understands the document structure, allowing you to set format template and automatically create a full table of contents.

Generic topics help pages

- Quick facts
- User access
- Cluster management
- Operational cluster control
- Interfaces overview
- Storage architecture
- Structural configuration
- PCI DSS encryption
- IBM Safer Payments security
- Revision control
- Automatic and assisted rule generation
- Benchmarking prevention performance
- Message computation

- Time representation
- Device identification

back to top

10.5.8 Levenshtein

IBM Safer Payments uses the Levenshtein algorithm to determine the closeness of two text values.

In information theory and computer science, the Levenshtein distance is a string metric for measuring the amount of difference between two sequences. The term edit distance is often used to refer specifically to Levenshtein distance.

The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character. It is named after Vladimir Levenshtein, who considered this distance in 1965.

The table below lists some exemplary Levenshtein distances.

Value 1	Value 2	Levenshtein distance
Godot	God0t	1
Müller	Mueller	2
Jon Jones	J. Jones	2
Pete Black	G. Peter Black	4
Marylin Monroe	MarylinManson	4
John Jakob Jones	John Jones	6
Chris Thomas	Nick Pye	11
A. Mueller	Mueller, Anton	10
A. Müller	Mueller, Anton	12

back to top

10.6 Definitions

This section explains some of the specific terminology used in the IBM Safer Payments documentation.

back to top

10.6.1 Service Consumer

When discussing integration of IBM Safer Payments with other systems, we use the following terminology:

- **Service provider**
  In all integration scenario, we refer to IBM Safer Payments as the "service provider", providing decision services to the service consumer.
- **Service consumer**
  We refer to any system that is connected to IBM Safer Payments as a "service consumer". This includes authorization systems, card management systems, automatic calling systems, etc. From the IBM Safer Payments server component point of view, also the IBM Safer Payments client (via the API) is a service consumer.

back to top