

RMA User Guide

CUSTOMER PRODUCT INFORMATION

Copyright

© Ericsson AB 2007–2021. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	User Interface Conventions	2
1.4	Typographic Conventions	2
2	Concepts	3
2.1	Ericsson Rule Engine	3
2.2	Rating Management Application	3
2.3	ERE Terminology	3
3	RMA Preferences	11
3.1	Prerequisites	11
3.2	Starting RMA	11
3.3	Optional Features	12
4	Rating Management Application Overview	17
4.1	Main Menu	18
4.2	RMA Settings	19
4.3	Toolbar	22
4.4	Status Bar	22
4.5	ERE Navigator	23
5	Working with the Service Definition	33
5.1	Creating a Service Definition	33
5.2	Opening a Service Definition for Editing	34
5.3	Using the Fields Definition Editor	34
5.4	Plug-In Editor	51
5.5	Constraints	60
5.6	Service Data Area	63
5.7	Saving the Service Definition	65
6	Working with Rating Periods	67
6.1	Creating a New Rating Period	68
6.2	Opening a Selection Tree for Editing	71



6.3	Selection Tree Control Commands	71
6.4	Common Selection Tree Input	77
6.5	Selection Tree Validation	77
6.6	Selection Tree Root	78
6.7	Special Input Editors	85
6.8	Definitions Element in the Selection Tree	98
6.9	Selection Tree Analysis	103
6.10	Using Nodes, Conditions, and Modifiers	104
6.11	Selection Tree Qualifiers	107
6.12	Saving the Selection Tree	109
6.13	Using the Bookmark Function	110
6.14	Using the Find Function	112
7	Testing and Simulating the Selection Tree	113
7.1	Simulation Case Management	114
7.2	Simulation Input Data	116
7.3	Simulation Procedure	120
7.4	Simulation Output	121
7.5	Simulation Sets	126
7.6	Simulation Case Storage	127
7.7	Selection Tree Test Tool	135
8	Plug-in Descriptions	143
8.1	Nodes	143
8.2	Conditions	145
8.3	Modifiers	183
9	ERE Tools	203
9.1	Import Wizard	203
9.2	Export Wizard	204
9.3	Distribution Wizard	206
9.4	Selection Tree Difference Detection Tool	208
9.5	Find Function	212
9.6	Branch Search	219
9.7	Multi Data Update Tool	222
10	Troubleshooting	229
10.1	Event Logging	229
10.2	Runtime Properties	232



10.3	Memory Use	232
10.4	Common Error Messages	233
11	Online Help	235
11.1	Online Help Navigation	235
11.2	Rating Manager Connection Input Fields	235
11.3	Rating Manager Input Field	236
12	Third Party License Agreements	237
12.1	Apache	237
12.2	ASM	240
	Glossary	243
	Reference List	245





1 Introduction

This user guide on Rating Management Application (RMA) is meant for the administrators of solutions containing the ERE framework. It describes how to create and maintain rule logic.

RMA is the part of the Ericsson Rule Engine (ERE) concept, which cannot function as a standalone product. To fully understand its context, it must be accompanied with information on an integrated solution or a product.

1.1 Purpose and Scope

The purpose of this user guide is to provide the common information for all solutions of RMA regardless where they are implemented. The document covers information how to manage definitions of services, which rating requires, including the following:

- Startup of the RMA GUI
- Description of the RMA GUI
 - ERE Navigator
 - Service Definition Editor
 - Selection Tree Editor
 - Simulation of selection trees

This user guide is also used as the online help text within the RMA application.

Additional operation and maintenance information is normally described in the documentation that comes with the solution or product where RMA is integrated, for example, a Service Implementation Description.

1.2 Target Groups

The target groups for this document are as follows:

- Application administrator
- System administrator

For more information regarding the target groups, refer to *SDP Customer Product Information Overview, Reference [1]*.



1.3 User Interface Conventions

The look and feel of the Graphical User Interface (GUI) may vary depending on the platform.

1.4 Typographic Conventions

This document follows a set of typographic rules that make the document consistent and easy to read. For more information about typographic conventions, refer to *SDP Customer Product Information Overview*, Reference [1].



2 Concepts

This chapter describes the terminology used when working with ERE.

2.1 Ericsson Rule Engine

ERE is a rule engine framework designed for integration in solutions. It is not a standalone rule engine, but rather provides components necessary to build one. When ERE is integrated into a solution, the behavior may differ somewhat from the behavior described in this document. The modified behavior is described in the documentation delivered with the specific solution, often in the form of a User Guide.

The rules used by the ERE are enclosed in selection trees. ERE processes input data based on logic in a selection tree, producing a resulting set of data, an example of which may be seen in Figure 1. The solution uses the resulting data to make a decision, such as deducting a rate for a service.

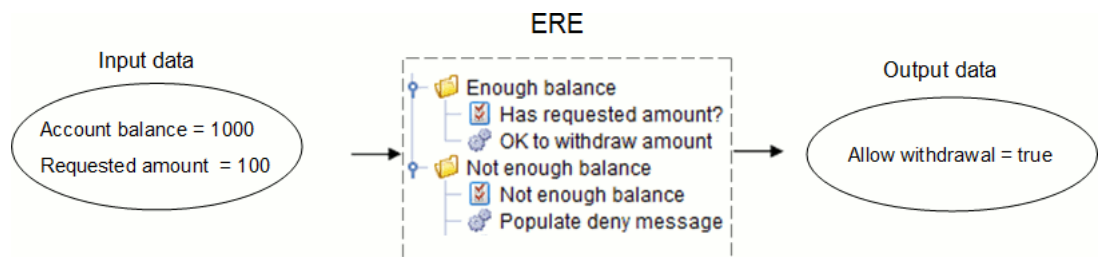


Figure 1 Example of Selection Tree Evaluation

2.2 Rating Management Application

RMA is a GUI for creating, configuring, and managing selection trees on one or more ERE servers. As with the ERE framework, RMA may be customized in many ways to meet solution-specific needs.

RMA automatically starts an ERE server on the local computer, called Internal ERE. The purpose of the Internal ERE is to act as a sandbox area where selection tree logic can be built and tested.

2.3 ERE Terminology

An ERE server contains at least one ERE manager, which contains selection trees, services, and related structural elements. Other terminology relevant to ERE may be found in the glossary. In Figure 2, an overview of ERE is given, and the different parts are described in the later sections.

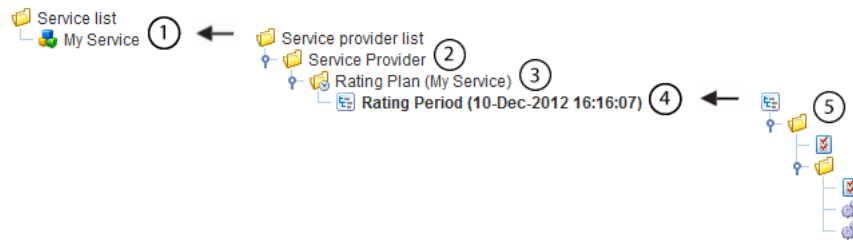


Figure 2 Overview of ERE

The following list is a legend to Figure 2

1. Service
2. Service Provider
3. Rating Plan
4. Rating Period
5. Selection Tree

2.3.1 Service

A **Service** defines all selection tree element types and definitions of the data fields that should be possible for use in an associated selection tree. The service may also contain additional rules, called constraints, which affect the behavior and configuration options in the elements. A service can be associated with one or more selection trees independent of which service provider they are contained in.

For information on how a service may be configured, see Section 5 on page 33.

2.3.2 Service Provider

A **Service Provider** holds one or more rating plans, which can be connected to the same or different services. The intent of this structural entity is to group rating plans with the same purpose. The service provider list holds all the service providers.

For information on how a service provider may be configured, see Section 4.5.3.3 on page 28.

2.3.3 Rating Plan

To execute a tree, the solution using ERE will select a **Rating Plan** and execute it with a specific time. A rating plan contains zero or more rating periods with unique start times. The start times creates a time line and only one rating period is valid for execution at any one time. This enables scheduling of the selection logic.

The rating periods contained in the rating plan must be associated with the same service and belongs to one service provider only.

A timeline of how rating periods in a rating plan are executed may be seen in Figure 3.

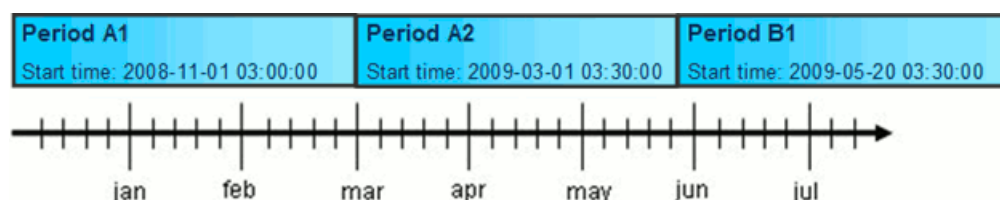


Figure 3 Rating Period Start Times Form a Timeline

For information on a rating plan may be configured, see Section 4.5.3.3 on page 28.

2.3.4 Rating Period

Rating Period holds the start time for one selection tree, and defines which selection tree is valid at the present time. It is not possible to define a stop time, so a rating period is valid until the start time of another rating period in the rating plan.

For information on a rating period may be configured, see Section 4.5.3.3 on page 28.

2.3.5 Selection Tree

A **Selection Tree** holds the rule logic for the solution.

It is built using elements called nodes, conditions and modifiers. The elements available in a selection tree, called plug ins in ERE terminology, are defined in the service, see Section 2.3.1 on page 4. A number of plug-ins are provided as part of the ERE framework and are described in this document. It is possible to create other plug-ins as part of the solutions which ERE is integrated into.

Elements at the same level in the tree are called sibling elements, while elements inside a node or sub-node are called child elements of that node.

The logic of the rules contained in the Selection Tree is changed by rearranging the elements or using other types of elements. The order in which the plug ins occur in the RMA GUI is the same as the order in which they are executed. An example of a selection tree may be seen in Figure 4.

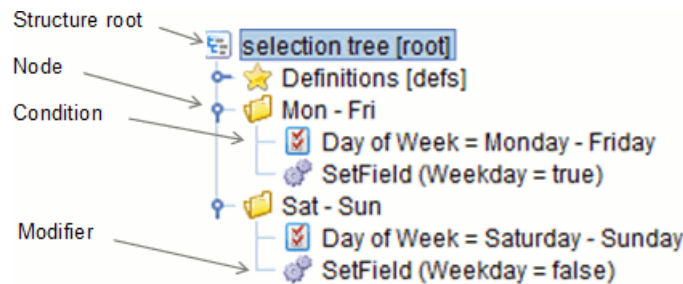


Figure 4 Selection Tree Elements

The **root** of the selection tree structure is the starting point for execution and holds configuration that is valid for the entire tree.

A **node** can contain other nodes, conditions and modifiers, and may define behavior for the contained children, for example, to invert the evaluation. The children contained in the node are executed from top to bottom and evaluation of the conditions under a node determines whether the child nodes or modifiers are executed or not. If a node has no conditions, its children are always executed. Under a node, conditions are placed first, whereas nodes and modifiers can be arranged in any order. The behavior of a node is described more thoroughly in Section 8.1 on page 143.

A **condition** is a logic decision point in the selection tree, and is used to construct boolean logic rules. Conditions may be divided into two types, base conditions and generic conditions. A base condition compares a configured value with the value of a pre-defined field. The field can be associated with the condition through the service or through the **FieldSelection** condition in the selection tree. See Section 8.2.3.9 on page 180 for more information on the **FieldSelection** condition. Generic conditions cannot be assigned to a field in a service. Section 8.2 on page 145 describes conditions and their use more thoroughly.

A **modifier** is the point in the selection tree where the calculations or the modifications of fields are performed. A modifier can use, modify or add data to fields defined in the service, depending on the **Parameter type** used for the field. See Section 5.3.6.1 on page 42 for the different parameter types. Some modifiers can also determine if the execution in the tree should stop or continue. Section 8.3 on page 183 describes modifiers and their use more thoroughly.

The **Definitions** element in the selection tree is a library where it is possible to define elements that can be used as link targets throughout the structure. For more details, see Section 6.8 on page 98.

2.3.6

Fields

The fields in ERE define data that is used when evaluating selection trees. The fields available to a selection tree are specified in the associated service.

There are two types of field definitions.



- Data field - A field that is defined with data type and parameter type. The parameter type defines if the field is used to store values, or provide values for other fields. The definition may either be set in the service, called a Service Defined Field (SDF), or in the selection tree, called a Tree Defined Field (TDF).
- Hierarchical field - A field with a hierarchical structure. The structure must end with at least one data field. All data fields within a hierarchical field structure must consist of the same parameter type, see Section 5.3.6.1 on page 42.

Fields can be defined to contain collections of representative data or grouped together with other fields. This makes handling data easier and more structured.

For more details on the different fields, see Section 5.3.2 on page 35.

2.3.6.1 Collections

Data fields, and hierarchical fields at top level, can be configured as collections. The different types of collections are described in the sections below.

2.3.6.1.1 NONE

Data field The field can contain a single value.

Hierarchical field The field can contain other hierarchical fields, data field collections, and data fields.

2.3.6.1.2 ARRAY

Data field The field is an array of values identified by a numeric index. The array can be defined with default values.

Hierarchical field The field is an array of indexed data fields. The field supports multiple data field collections.

Each array entry consists of:

- A numeric index.
- One or several data fields, and data field collections in the hierarchical structure.

2.3.6.1.3 MAP

Data field The field value is a collection of value pairs. One pair consists of a string key and its corresponding value. Within one collection, all keys must be unique.

**Hierarchical field**

The field is a map of data fields. The field supports multiple data field collections.

Each map consists of:

- Key: One or several key data fields in the hierarchical structure.
- Value: One or several non-key data fields, and data field collections, in the hierarchical structure.

At least one child data field must be set as key field in the hierarchical field definition. If more than one key is selected for the map collection, these fields are combined to create a composite key for the map collection.

2.3.6.1.4**SET****Data field**

The field value is a collection of unordered unique values.

Hierarchical field

The field is an unordered collection of data fields.

Each set entry consists of all data fields in the hierarchical structure. The value, or combined value, of these fields must be unique.

2.3.7**Groups**

A group assembles related data fields, for easier access in the RMA simulation. The different types of groups are described in the sections below.

2.3.7.1**Group**

A group may contain other groups and data fields.

2.3.7.2**Array Group**

An array group is a collection of equally sized array fields. The array fields make up a two-dimensional array, where each row is treated as a separate data structure. Each row must have a unique integer identifier.

It is possible to mix the data types and parameter types for the array fields within the array group, but it is not possible to have an array group without any defined fields. An array group may not contain other groups.



2.3.7.3

Map Group

A map group is a collection of map fields. In a map group, each row is treated as a separate data structure. Each row must have an identifier, known as the key field for the row. This may be defined as any field within the row, or a combination of fields to create a composite key that better defines the row in the data structure. Fields with parameter types WORKING or VALIDATION cannot be chosen as key field. See Section 5.3.6.1 on page 42 for the different parameter types.

It is possible to mix the data types and parameter types for the fields within the map group, but it is not possible to have a map group without any defined fields. A map group may not contain other groups.





3 RMA Preferences

RMA can be started as a standalone product. Other ways of starting RMA may exist depending on the solution or product where RMA is integrated.

The RMA can be installed and executed on any operating system supporting Java 1.8 or later.

3.1 Prerequisites

Conditions that must apply before this task can be completed are as follows:

- Java version 1.8 or later version, has to be installed.

If the international version of Java Runtime Environment (JRE) is used, the language used in some of the file access dialogs use the settings for the local language. If the English version of Java Runtime Environment is used, all local settings are ignored.

3.1.1 Checking Java Version

Type **java -version** at a console, or the command prompt in Windows. See Example 1.

```
# java -version
```

Example 1 Input at a Console or Command Prompt

Output from the command is similar to the following:

```
java version "1.8.0_162"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_162-b12)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 25.162-b12, mixed mode)
```

Java can be downloaded from <http://www.oracle.com>.

3.2 Starting RMA

To start RMA, do as follows:

For Windows:

- Locate the **bin** directory in the directory where RMA has been installed, and from the command line enter **startup.bat**, or open the file in the explorer window.

For Unix:

- The first time RMA is started, locate the directory where RMA has been installed, and from the command line enter `chmod +x bin/startup.sh`.
- If RMA has been started once using the command described in the previous bullet, then locate the directory where RMA has been installed, and from the command line enter `./bin/startup.sh`.

3.2.1 Starting RMA for the First Time

The first time RMA is started, the user is prompted with the information displayed in Figure 5.

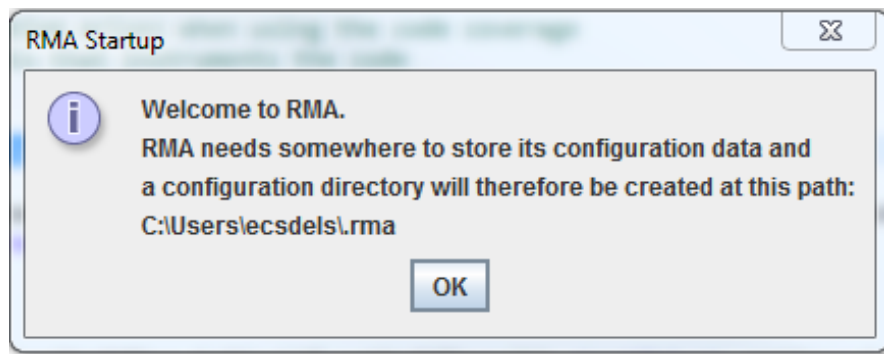


Figure 5 RMA Startup

Selecting the **OK** button causes RMA to create an RMA home directory where the RMA configuration is stored.

The internal Rating Manager is provided with default settings. These settings may be changed if required.

1. Right-click on the **Rating Manager ERE (Internal ERE)** text and choose **Properties** from the context-sensitive menu.

The properties window for the internal Rating Manager server is shown.

2. Change the required data.
3. Press **OK**.

Note: If the user wants to connect another Rating Manager file that contains already designed services; press the ... button. Locate the Rating Manager file and press **OK**.

3.3 Optional Features

The behavior of some functions listed in this document depends on optional features that need to be activated in ERE. These optional features vary on installations and need to be requested by the customer.



3.3.1 Default Active Features

In a default installation of ERE, the features in Table 1 are active.

Table 1 Default Active Features

Name	Function	Details
NEW_RATING_PERIOD	Function to disable or enable the possibility to create a new rating period in a specific node. "New Rating Period" will be visible in the menu if enabled.	See Section 6.1 on page 68.
DELETE_RATING_PERIOD	Function to disable or enable the possibility to delete a rating period in a specific node. "Delete" will be visible in the menu if enabled.	See Section 6.3 on page 71.
CHANGE_RATING_PERIOD_PROPERTIES	Function to disable or enable the possibility to change the properties for a rating period in a specific node.	See Section 6.1 on page 68.
SAVE_SELECTION_TREE	Function to disable or enable the possibility to save the selection trees in a specific node. "Save" will be visible in the menu in the selection tree if enabled.	See Section 6.3 on page 71.
SHOW_SERVICE_LIST	Function to disable or enable the possibility to see the service list and its services in a specific node.	See Section 4.5.3.1 on page 27.
CONSTRAINTS	When the function is disabled the constraints tab in the service (in a specific node) will not be shown, if there are no constraints added already. If there are constraints added or if the function is enabled the constraints tab will be shown.	See Section 5.5 on page 60.
IMPORT_ALLOWED	Function to disable or enable the possibility to import to a specific node.	See Section 9.1 on page 203.
DISTRIBUTION_TARGET	Function to disable or enable the possibility to distribute a rating period to a specific node.	See Section 9.3 on page 206.



3.3.2 Default Inactive Features

In a default installation of ERE, the features in Table 2 are inactive.

Table 2 Default Inactive Features

Name	Function	Details
DEBUG	Function to disable or enable the possibility to jump to non-active rating periods in the simulation tab in a specific node.	See Section 7.1 on page 114.
ALLOW_SERVICEPROVIDER_LIST_MENU	Function to disable or enable the possibility to view the service provider list menu in a specific node.	See Section 5.3.3.1 on page 37.
TEST_GENERATION_TOOL	Function to disable or enable the possibility to generate a simulation set from a modifier in the selection trees in a specific node.	See Section 7.7 on page 135.
SPEED_TYPING	Function to disable or enable the possibility to use speed typing in the selection trees in a specific node. When it is enabled and a new node is created its name gets focused directly.	See Section 6.3.4 on page 76.
PASTE_SPECIAL	Function to disable or enable the possibility to use “Paste...”, which allows the user to choose between “Paste as child” or “Paste as sibling” in the selection trees in a specific node.	Section 6.3 on page 71
BOOKMARKS	Function to disable or enable the possibility to see and create new bookmarks in the selection trees in a specific node.	See Section 6.13 on page 109.
BRANCH_SEARCH	Function to disable or enable the possibility to search for branches in the selection trees in a specific node. “Search for branches...” is visible in the menu if enabled.	See Section 9.6 on page 219.



Table 2 Default Inactive Features

SHOW_FIND_PARAMETERS	Function to disable or enable the possibility to show and find parameters in the selection trees in a specific node.	See Section 9.5.4 on page 218.
MULTI_DATA_UPDATE	Function to disable or enable the possibility to use multi-data update in a specific node, which is found in a rating period's menu, if enabled.	See Section 9.7 on page 222.
SIMULATION_PANEL_SEARCH	Function to disable or enable the possibility to search in the simulation panel in a specific node.	See Section 7 on page 113.





4 Rating Management Application Overview

RMA has tools for defining, building, testing, and distributing rating manager configuration to different ERE instances. It contains an internal ERE and other defined ERE instances.

An example of the RMA GUI can be seen in Figure 6 and is described in Table 3.

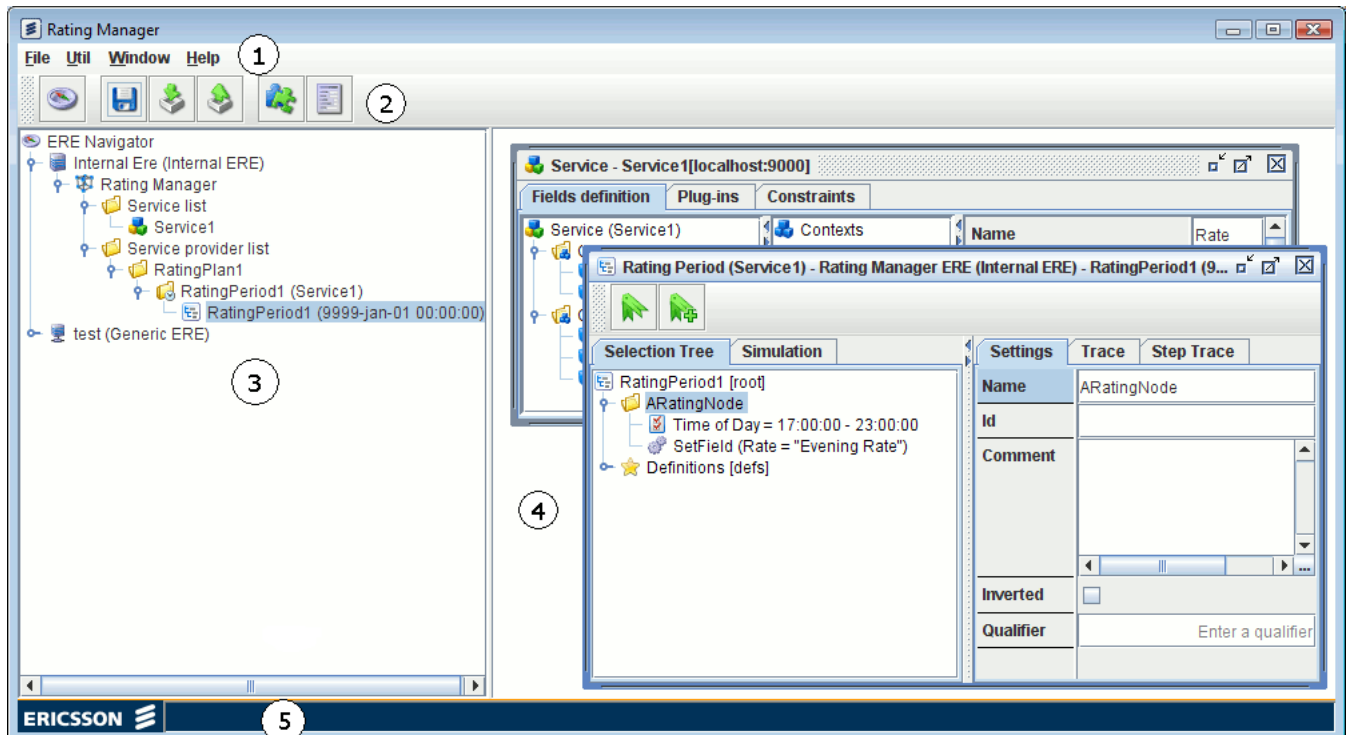


Figure 6 RMA GUI Example

Table 3 RMA GUI

	GUI Area	Description
1	Menubar	Contains menu options for RMA. For more information, see Section 4.1 on page 18.
2	Toolbar	Contains shortcuts to commonly used operations in RMA. For more information, see Section 4.3 on page 22.
3	ERE Navigator	Contains an overview of the defined ERE instances. For more information, see Section 4.5 on page 23.



Table 3 RMA GUI

	GUI Area	Description
4	RMA Desktop	Contains editors and tools for working with rating manager configurations.
5	Status Bar	Provides important information such as errors and warnings. For more information, see Section 4.4 on page 22.

4.1 Main Menu

The main menu contains the options described in Table 4:

Table 4 Options in the Main Menu

Menu Option	Description
File	<ul style="list-style-type: none">• Save - Saves the information in the current window in RMA desktop.• Import - Provides the possibility to import rating manager configurations into the navigator. Opens the import wizard, see Section 9.1 on page 203.• Export - Provides the possibility to export rating manager configurations from the navigator. Opens the export wizard, see Section 9.2 on page 204.• ERE Navigator - Hides and shows the ERE Navigator.• Settings - Provides the possibility to customize RMA settings. See Section 4.2 on page 19 for configurable options.• Exit - Exits RMA.
Util	<ul style="list-style-type: none">• Log - Opens the log window. For more information, see Section 10.1 on page 229.• Selection Tree Difference Detection Tool - Opens the Selection Tree Difference Detection Tool, see Section 9.4 on page 208• Simulation Database Configuration - Opens the database configuration dialog, see Section 7.6.2 on page 127 <p>This menu can also contain entries specific to the application in which ERE is integrated. See Section 2 on page 3 for more information.</p>



Table 4 Options in the Main Menu

Menu Option	Description
Window	<p>The menu contains a list of all application windows which are currently open. Select an application from the list to make it the active window.</p> <ul style="list-style-type: none"> • Close Inactive Windows - Closes all the windows in the RMA desktop, except for the one which is currently active. • Close All Windows - Closes all windows in the RMA desktop. • Cascade - Displays all open windows in RMA desktop, in a cascade. • Tile - Displays all open windows in RMA desktop, side by side. • Save Window Positions - Saves the positions of the open windows. The next time RMA is opened, the saved window positions are used.
Help	<ul style="list-style-type: none"> • About - Shows application version and copyright information. It also provides the RMA runtime properties. For more information regarding runtime properties, see Section 10.2 on page 232. • RMA - User Guide - Opens the online help window. For more information, see Section 11 on page 235. <p>This menu can also contain help files specific to the application in which ERE is integrated. See Section 2 on page 3 for more information.</p>

4.2 RMA Settings

The **RMA Settings** dialog box is accessed through the main menu > **File** > **Settings**.

The **RMA Settings** dialog box includes settings for:

- **General**
- **ERE Navigator**
- **Log Settings**
- **Selection Tree Editor**

4.2.1 General

The following settings are configurable:



Table 5 General Settings

Option	Explanation	Default Value
Sound	Turn RMA sounds on or off.	On
Show Memory Usage	Turn exposure of RMA memory usage bar on or off. See Section 10.3 on page 232 for more information on memory use.	Off

4.2.2 ERE Navigator

The following settings are configurable:

Table 6 ERE Navigator Settings

Option	Explanation	Default Value
Period sorting	Sort rating periods by name or date in the ERE navigator.	Name sorting
Highlight current rating period	Turn indication of all active currently used rating periods on or off.	Off

4.2.3 Log Settings

The following settings are configurable:



Table 7 Log Settings

Option	Explanation	Default Value
Log	Adjustable value for maximum entries in the log. Allowed value range: 0 - 300.	300
	Adjustable minimum log level: <ul style="list-style-type: none"> • Finest • Finer • Fine • Config • Info • Warning • Severe • Off See Section 10.1.2 on page 230 for more information on log settings.	Finest
Log File	Selectable location and format for the log file. Select the Use Persistent Log File check box to write the log file to a specified location. Selectable output for the log file can be one of the following formats: <ul style="list-style-type: none"> • XML • Plain 	No log file selected

4.2.4 Selection Tree Editor

The following settings are configurable:

Table 8 Selection Tree Editor Settings

Option	Explanation	Default Value
Use local clipboard	Turn the use of local clipboard on or off. If off is selected, the internal RMA clipboard is used. This allows XML structures to be copied from an object in the Selection Tree Editor.	On









Table 8 Selection Tree Editor Settings

Show only executed branches in step trace	Turn hiding of non-executed branches, in the step trace output of a simulation, on or off. If on is selected, only executed branches and objects on the same level are shown in the step trace output. Child objects of non-executed branches are hidden. See Section 7.4.1 on page 121 for more information on simulation step trace.	Off
Confirm Drag&Drop on move	Turn confirmation of drag-and-drop move operations on or off. This setting applies only in the Selection Tree Editor. See Section 6.3.3 on page 76 for more information on drag-and-drop commands.	Off

4.3 Toolbar

The RMA toolbar is described in Table 9.

Table 9 Toolbar Button Description

Symbol	Description
	ERE Navigator Shows or hides the ERE Navigator.
	Save Saves the content in the currently active window, in RMA desktop.
	Import Imports a rating manager configuration, see Section 9.1 on page 203.
	Export Exports the selected rating manager configuration to a file, Section 9.2 on page 204.
	Selection Tree Difference Detection Tool. Starts the Selection Tree Difference Detection Tool, see Section 9.4 on page 208.
	Log Shows the log application, see Section 10.1 on page 229.

4.4 Status Bar

The status bar is found at the lower left of the RMA GUI and is used to inform the user of the result of input to RMA. An example of a displayed message in the status bar is shown in Figure 7.



Figure 7 Example of Status Message

4.5 ERE Navigator

The ERE navigator is the starting point for RMA. From the navigator, it is possible to connect to and work with ERE instances. It is possible to hide the navigator, see Table 9.

The RMA desktop is used as a work area to perform different tasks such as edit rating periods, perform simulations and so on.

To perform an action in the ERE navigator, right-click on the navigator element and choose from the menu. The popup menu is context-sensitive so the menu options that are shown depends on which tree element that has been selected. Those context-sensitive menus and property windows are described in the following sections of this chapter.

Figure 8 shows an example of an ERE navigator with internal and remote ERE instances.

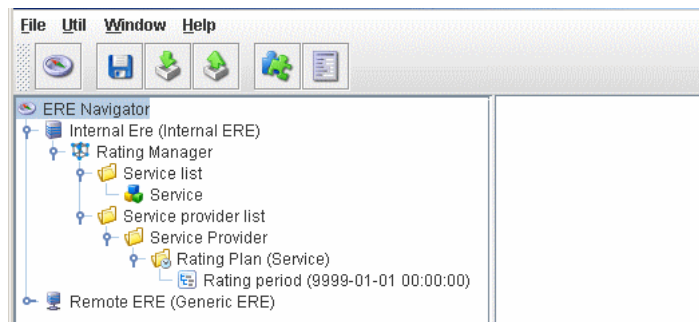


Figure 8 RMA Window with ERE Navigator Visible

4.5.1 Icons in ERE Navigator

In the menu there are some different icons which indicate the status on the ERE instances in RMA. The icons in the navigator are explained in Table 10.

Table 10 Icons in Rating Manager Connection Menu

Symbol	Description
	Indicates an online internal ERE instance.
	Indicates an offline internal ERE instance.
	Indicates an online remote ERE instance.
	Indicates an offline remote ERE instance.



4.5.2 ERE Navigator Menu

ERE Navigator connection menu is available by right-clicking on the ERE Navigator root icon or on any of the defined ERE instances. Which menu options that are shown, depends on which tree element in the ERE Navigator that is selected.

4.5.2.1 Adding an ERE Node

ERE instances are added to the navigator tree by right-clicking on the ERE Navigator icon at the root of the selection tree, see Figure 9.

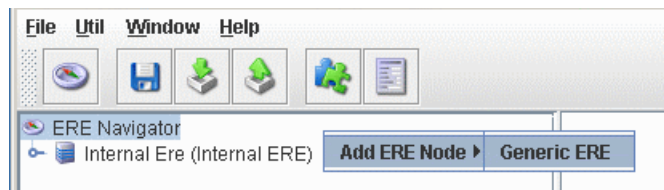


Figure 9 Adding an ERE Node

In the example in Figure 9, a generic ERE node is added. The application in which ERE is implemented may have several standard types of ERE instance defined, which may be selected in a similar fashion. When clicking on one of the choices a new window opens, see Figure 10. The parameters vary depending on the ERE instance type and are described in Table 11.

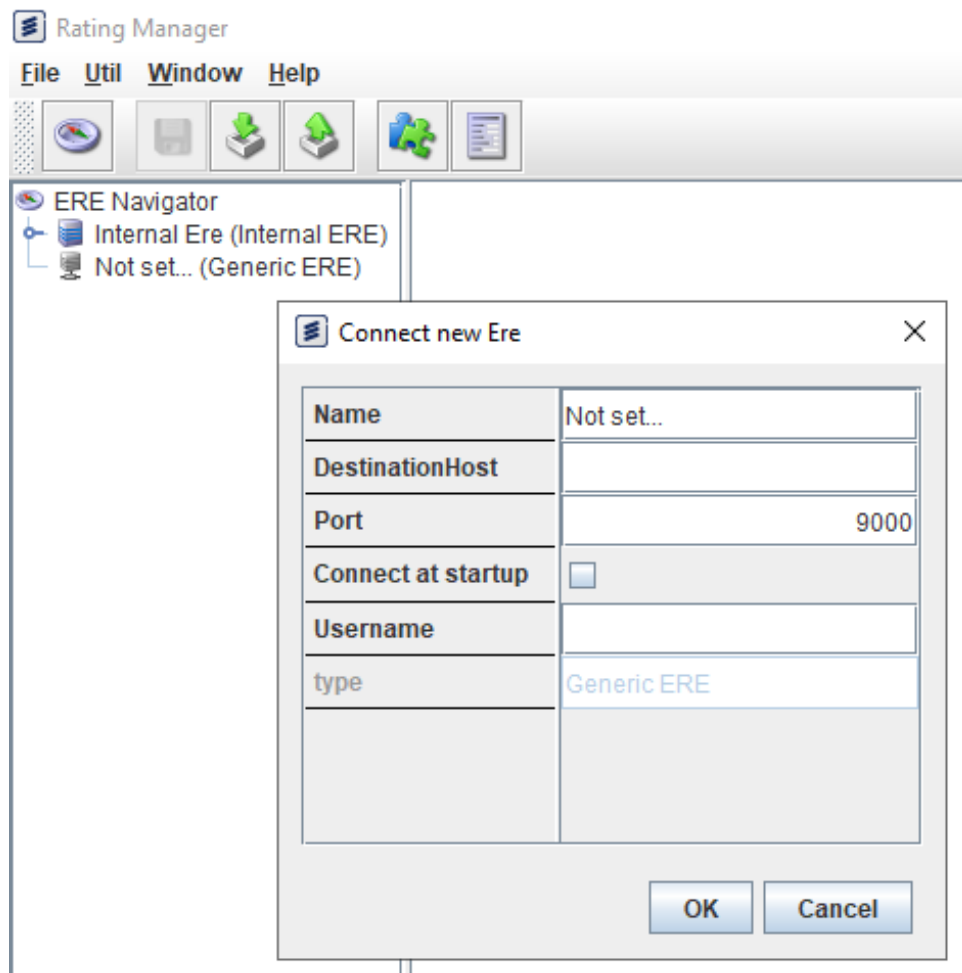


Figure 10 Adding an ERE Instance

Table 11 Parameters when Adding an ERE Instance

Menu option	Description
Name	Host name.
Port	The port number at the remote host. Default port number is 9000.
Username	The user name for the user when connecting to the server.
Password	The password for the user when connecting to the server. Note: Applicable for Internal ERE only.
Enabled	Controls if the ERE instance is enabled in RMA.
Description	Free text field.
Connect at startup	RMA tries to establish a connection to the remote host by default when ERE is started.



Table 11 Parameters when Adding an ERE Instance

Menu option	Description
DestinationHost	Address for the remote host.
type	A read-only field containing the type of ERE instance.
Rating Manager File	Points to an XML file where the configuration of the Internal ERE is stored. It is possible to browse to this file.

4.5.2.2 Connecting to an ERE Instance

To connect to an ERE node, double-click on the selected node name. When an ERE is connected, the navigator is updated with the current rating manager configuration. If changes are made to the ERE instance by another user, they are only visible in the current users RMA after a refresh is performed.

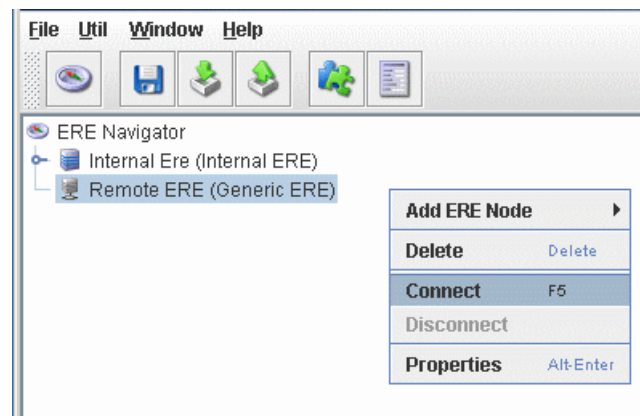


Figure 11 Connecting a Node

Table 12 Connecting to Node

Menu option	Description
Add ERE node	See Section 4.5.2.1 on page 24.
Delete	Deletes the node from the navigator.
Connect	Establishes a connection to the selected ERE instance.
Refresh	Retrieves the latest rating manager configuration from the server.
Disconnect	Disconnects the selected ERE instance.
Properties	Opens a properties window the connection data for the ERE instance can be modified.



4.5.3 Rating Manager Menu

All elements in the ERE Navigator have menu options, which appear in a drop-down menu when right-clicking on the element. Some options are common to all elements, while some relate to a specific element.

The options of the **Rating Manager** menu are described below, in Table 13.

Table 13 Rating Manager Menu Options

Menu Option	Description
Import...	Opens a wizard to import an entire rating manager configuration from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export a rating manager configuration to file, see Section 9.2 on page 204.

4.5.3.1 Service List Menu

A service may be added to an ERE instance by selecting **New Service** from the drop-down menu, when right-clicking on the **Service list**. The menu options are shown in Figure 12 and described in Table 14.

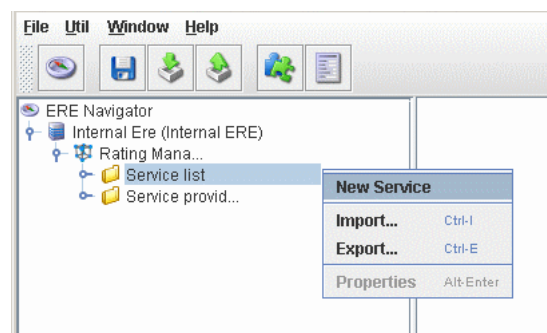


Figure 12 Service List Menu

Table 14 Service List Menu Options

Menu Option	Description
New Service	Opens a dialog where a new service can be defined.
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export services to file, where multiple services can be selected for export, see Section 9.2 on page 204.
Properties	Not available for the service list. For information regarding the properties of a service, see Table 21.



4.5.3.2 Service Menu

The service menu is displayed when right-clicking on a service. The menu options are described in Table 15.

Table 15 Service Menu Options

Menu Option	Description
New Service	Opens a dialog where a new service can be defined.
Delete	<ul style="list-style-type: none">• From this ERE - Deletes the selected service from location where the action is performed.• From multiple EREs - Opens up the ERE Distribution wizard, where the service can be deleted from multiple EREs.
Edit Service Definition	Opens the Service editor.
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export the selected service to file, see Section 9.2 on page 204.
Distribute...	Opens the ERE Distribution wizard to distribute a service to several EREs, see Section 9.3 on page 206.
Properties	Shows a dialog window with: <ul style="list-style-type: none">• Service definition file• Name• Service condition file

4.5.3.3 Service Provider List Menu

A rating period may be added to an ERE instance by selecting **New Rating Period** from the menu by right-clicking on the **Service Provider list**. The menu options are shown in Figure 13 and described in Table 16.

The service provider and rating plan are also defined when a rating period is created.

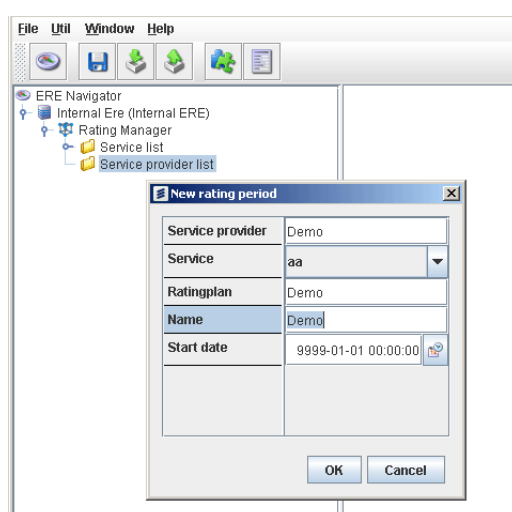


Figure 13 Selecting Rating Plan and Rating Period, when Adding Rating Period from Service Provider

Table 16 Service Provider List Menu Options

Menu Option	Description
New Rating Period	Opens a dialog where a new rating period can be defined.
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export a rating period list to file, where multiple rating periods can be selected for multiple rating plans. The related services can optionally be selected for export in the operation, see Section 9.2 on page 204.

4.5.3.4 Service Provider Menu

The service provider menu is displayed when right-clicking on a service provider. The menu options are described in Table 17.

Table 17 Service Provider Menu Options

Menu Option	Description
New Rating Period	Opens a dialog where a new rating period can be defined.
Delete	<ul style="list-style-type: none"> • From this ERE - Deletes the selected rating plan and all its content from the location where the action is performed. • From multiple EREs - This option is only available when at least one rating period is selected.
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export a rating period list to file, where rating periods contained in the rating plan can be selected. The related service can optionally be selected for export in the operation, see Section 9.2 on page 204.



4.5.3.5 Rating Plan Menu

The rating plan menu is displayed when right-clicking on a rating plan. The menu options are described in Table 18.

Table 18 Rating Plan Menu Options

Menu Option	Description
New Rating Period	Opens a dialog where a new rating period can be defined.
Filter	Filter rating periods of the rating plan according to name, see Figure 14.
Delete	<ul style="list-style-type: none">• From this ERE - Deletes the selected rating plan and all its content from the location where the action is performed.• From multiple EREs - This option is only available when at least one rating period is selected.
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export a rating period list to file, where rating periods contained in the rating plan can be selected. The related service can optionally be selected for export in the operation, see Section 9.2 on page 204.

4.5.3.5.1 Using Filter in a Rating Plan

The rating plan node contains a filter function. This is used when there is a large number of rating periods in one rating plan, making it difficult to browse. The filter takes a case-sensitive input string which is matched against the names of the rating periods in the node. Only the rating periods with names containing the input string are then shown in the rating plan node.

To configure the filter, right-click on a **Rating Plan** node and choose **Filter** from the context-sensitive menu.

A dialog appears where the filter string can be typed.

Rating plans with configured filters are marked with a different icon in the navigator, see Figure 14.

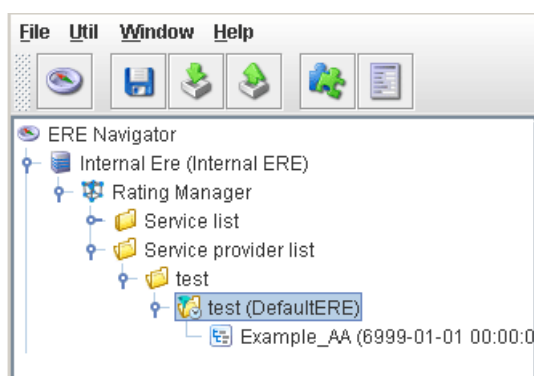


Figure 14 Filtered Rating Plan

The filter is removed by opening the filter dialog and erasing the filter string.

4.5.3.6 Rating Period Menu

The rating period menu is displayed when right-clicking on a rating period. The menu options are described in Table 19.

Table 19 Rating Period Menu Options

Menu Option	Description
New Rating Period	Opens a dialog where a new rating period can be defined, see Table 20.
Delete	<ul style="list-style-type: none"> • From this ERE - Deletes the selected rating plan and all its content only from the location where the action is performed. Multiple selected elements within the same ERE node can be deleted. • From multiple EREs - This option opens a wizard with possibility for deleting the selected element from several ERE instances. Multiple selected elements cannot be deleted from multiple EREs.
Edit	<p>Opens an editor window for the rating period.</p> <p>For more information regarding how to work with selection trees, see Section 6 on page 67</p>
Import...	Opens a wizard to import from file, see Section 9.1 on page 203.
Export...	Opens a wizard to export the selected rating period to file. The related service can optionally be selected for export in the operation, see Section 9.2 on page 204.
Distribute...	Opens a wizard to distribute a rating period to several EREs, see Section 9.3 on page 206.



Table 19 Rating Period Menu Options

Menu Option	Description
Compare to...	Opens the Selection Tree Difference Detection Tool . For more information about the tool see Section 9.4 on page 208.
Properties	Opens a dialog where Name and Start date of the rating period may be modified. The Rating Tree file is also displayed for the rating period. It points to the file where the Rating Period configuration is saved.

Adding a new rating period brings up a dialog window described in Table 20.

Table 20 Properties in New Rating Period Dialog

Menu Options	Description
Service Provider	The name for the new service provider.
Service	Drop-down menu that allows the user to select between available services.
Rating plan	Set a name for the rating plan.
Name	The desired name for the rating period.
Start date	Day and time from when the rating period is valid. The start date must be unique within the rating plan.

Selected rating period can be deleted by using the delete button on the keyboard. Using the right mouse button on the selected rating periods will bring up the rating manager menu options from where deletion can be made according to the selected option. To select multiple rating periods hold down **Ctrl + left mouse button**. Multiple selection is possible only for non-active rating periods within the same ERE node. When multiple rating periods have been selected, it is only possible to delete them from that ERE node.

Drag-and-drop operations can be made to automatically start an export and import operation of a rating period to a target selected, in a single operation. Only rating periods within the same service can be exported and imported with this single drag-and-drop operation. Both within an ERE node and between ERE nodes. The ERE wizard shows the results of the operation. For more information on export operations, see Section 9.2 on page 204. For more information on import operations, see Section 9.1 on page 203.



5 Working with the Service Definition

The service definition is a repository containing declarations for the building blocks possible to use when working with a selection tree. See Section 2 on page 3 for further explanation of the service definition concept.

The service editor makes it possible to define a service.

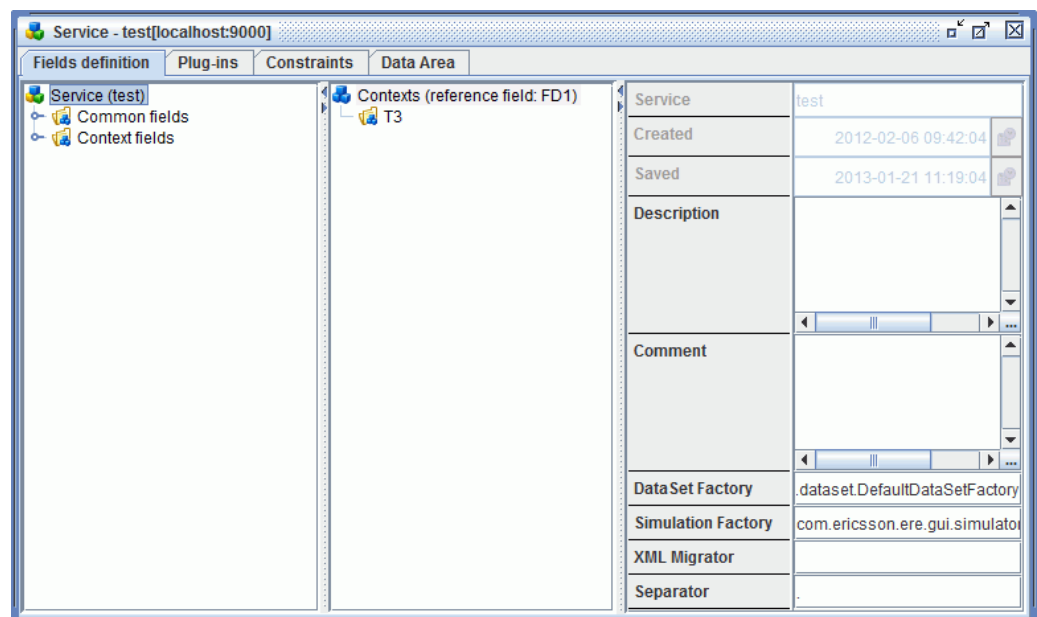


Figure 15 Service Editor

The editor consists of the following four tabs:

- **Fields definition** - Area to define fields and conditions. There is also a definition area for **Contexts**, see Section 5.3 on page 34 for more details.
- **Plug-ins** - Area to define plug-ins, see Section 5.4 on page 51 for more details.
- **Constraints** - Area to enable fields, conditions, modifiers, operations and features to be included or excluded from the RMA GUI. See Section 5.5 on page 60 for more details.
- **Data Area** - Area to define data for external data sources, see Section 5.6 on page 63 for more details.

5.1 Creating a Service Definition

A new service definition can be created in the ERE Navigator by right-clicking **Service List** in the navigator tree, and choosing **New Service** from the context-sensitive menu.

A **New Service** dialog window appears where the name of the new service can be entered.

All existing services are found in the **Service list**.

5.2 Opening a Service Definition for Editing

All services which have been created may be opened for viewing and editing by double-clicking on the appropriate service in the ERE-Navigation Tree. It is also possible to right-click on a service and choose **Edit Service Definition** in the context-sensitive menu.

5.3 Using the Fields Definition Editor

This editor is shown when choosing the **Fields definition** tab in the Service editor. The window is structured with two tree structures and a configuration area, see Figure 15. The left tree contains the **Fields definition** and the right tree contains the **Contexts**. The configuration area shows the properties for the element currently selected in a tree.

5.3.1 Editing the Service

The properties of the service are shown in the configuration area to the right, see Figure 16.

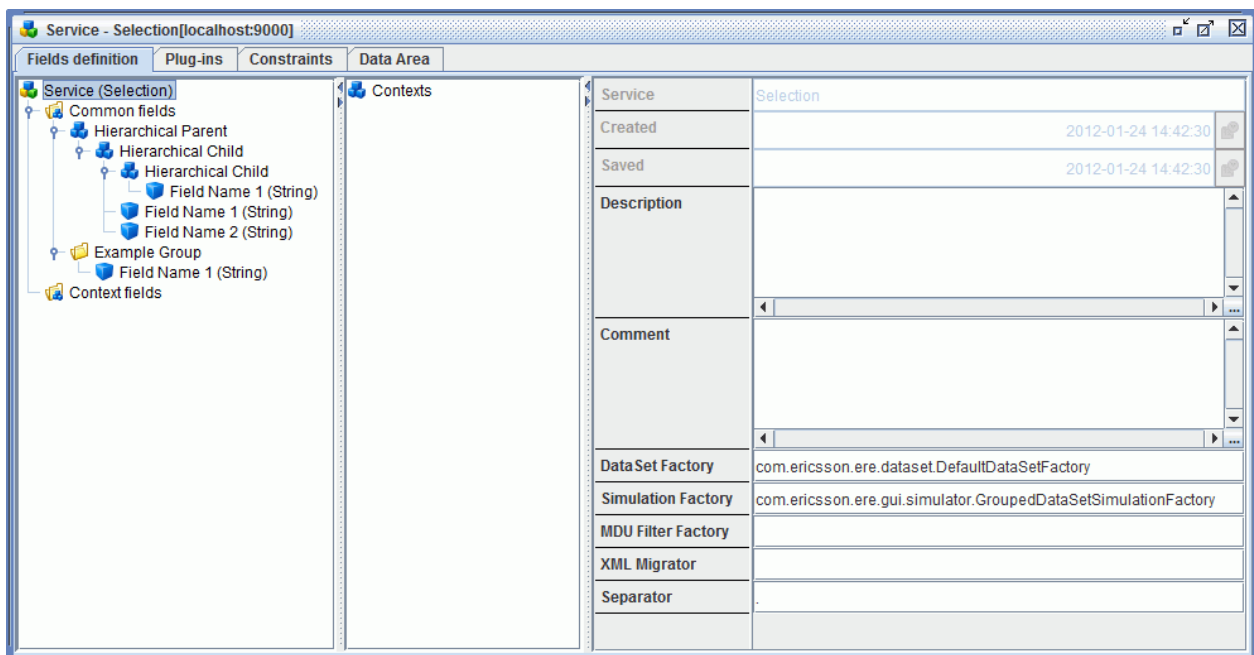


Figure 16 Editing the Service Element



These properties are described in Table 21.

Table 21 Properties for the Service

Property	Description
Service	The name of the Service. This property is read-only.
Created	The date and time when the Service Definition was first created. This parameter is read-only.
Saved	The date and time when the Service Definition was last saved. This parameter is read-only.
Description	An optional description of the Service Definition. Used to document the purpose of the service definition.
Comment	An optional comment for the Service Definition.
DataSet Factory	The class name of the DataSet Factory to be used for this Service Definition, if a specific DataSet instance should be used. If the field is empty com.ericsson.ere.dataset.LegacyDataSetfactory will be used.
Simulation Factory	The class name of the Simulation Factory to be used for this Service Definition, if a specific Simulation Factory instance should be used. If the field is empty ericsson.ere.simulation.SimulatorPanel will be used.
MDU Filter Factory	An optional customization point for Multi Data Update, enabling custom filtering of what elements in the rating period that is available for editing. When used, the text field contains the name of the class that does the element filtering. See the programmer's guide for more information.
XML Migrator	An optional customization point for automatic update (migration) of rating periods when they are loaded. When used, the text field contains the name of the class that does the load time migration. See the programmer's guide for more information.
Separator	Configurable separator for hierarchical fields. Separates parent field names from child field names in hierarchical structures.

5.3.2 Fields

Fields can be defined in the **Fields definition** tree. Such fields are called service defined fields (SDF). An SDF may be defined by adding, editing and deleting elements. These elements are described as follows:

- Data field - A value that can be used as an input- or output parameter in a selection tree. Conditions can be connected to data fields. The defined conditions can be used in the selection tree to decide which path the execution should take. Data fields can be grouped together. These groups are also used when the data fields are shown in the Simulation panel, see Figure 104. For more information on the different actions, see Section 5.3.3.1 on page 37.
- Hierarchical field - A field that consists of at least one data field in a hierarchical structure.

The characteristics and restrictions of a hierarchical field are listed below:

- Unique field names at the same level within the structure.
- Hierarchical fields can be set to a collection at top level.
- Hierarchical fields can contain other hierarchical fields, data field collections, and data fields.
- A hierarchical field collection cannot contain other hierarchical fields. Only data fields and data field collections are supported.
- All data fields within a hierarchical field structure must consist of the same parameter type, see Section 5.3.6.1 on page 42.
- All hierarchical fields must consist of at least one data field.

An example of a hierarchical structure is displayed in Figure 17. The configured separator is . and the name scope for the highlighted data field is User . Name . ID. As displayed in the picture, the same name can be used for multiple fields if they are on different levels in the structure.

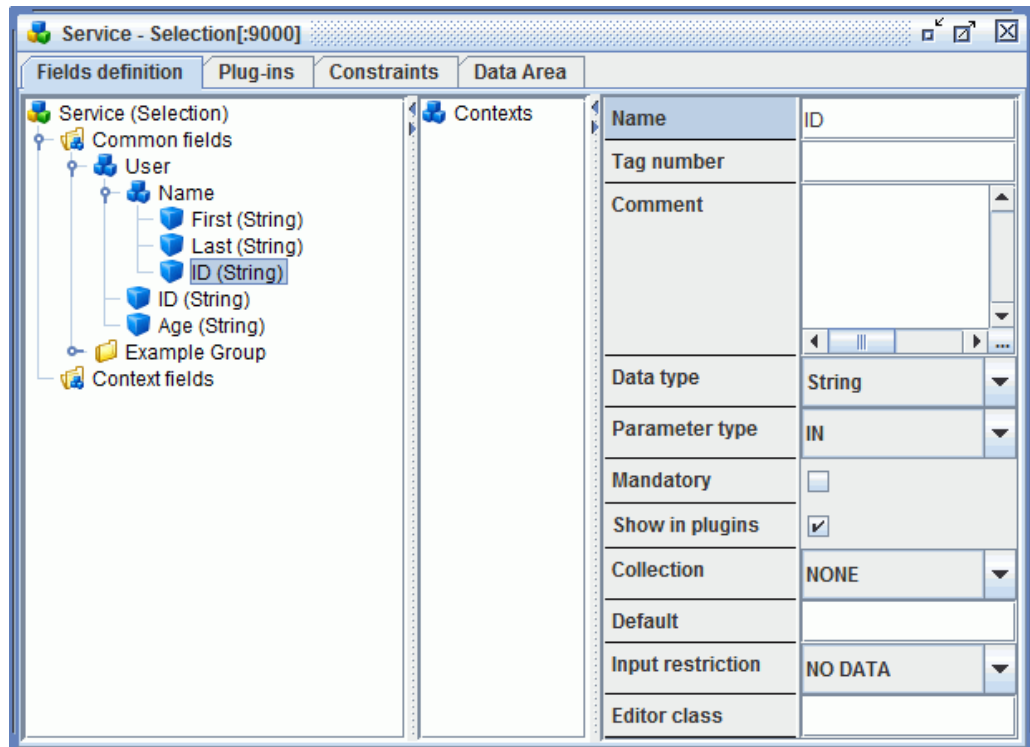


Figure 17 Hierarchical Field Structure

Section 5.3.3.1 on page 37 describes how an SDF may be edited.



5.3.3 Editing the Field Definition

The definition can be edited using the context-sensitive menu or by pressing the key combination connected to an action, see Table 22. The menu is accessed by right-clicking anywhere in a tree structure. The menu offers different editing alternatives depending on what kind of tree element that has been selected, see Figure 18.

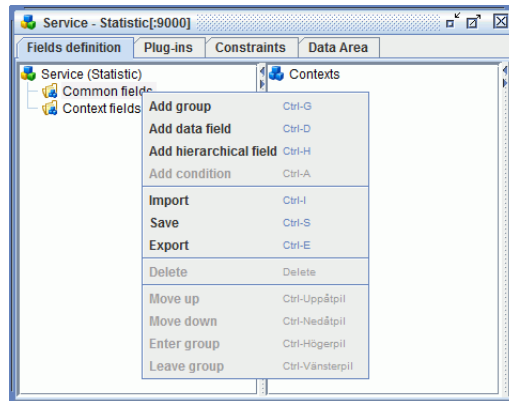


Figure 18 The Context-Sensitive Menu

5.3.3.1 Context-Sensitive Menu

Table 22 explains which actions that are available for the different element types and the corresponding key combination.

All alternatives are not applicable for every element type.

Table 22 Service Definition Menu Alternatives

Action	Description	Hot key
Add group	Adds a new group to the definition.	CTRL+G
Add data field	Adds a new data field to the definition. The new data field is placed below the selected element in the tree or, if a group is selected, in the selected group. Only available in the Fields tree.	CTRL+D
Add hierarchical field	Adds a new hierarchical field to the definition. The new hierarchical field is placed below the selected element in the tree. Only available in the Fields tree.	CTRL+H
Add condition	Adds a new condition and connects it to the selected data field. Only available in the Fields tree.	CTRL+A
Add context	Adds a new context, which data field or group links can be grouped into. Only available in the Context tree.	CTRL+N
Import	Opens an import wizard where a file to import can be chosen. Read more about the import function in Section 9.1 on page 203.	CTRL+I



Table 22 Service Definition Menu Alternatives

Action	Description	Hot key
Save	Saves the current Service definition.	CTRL+S
Export	Opens the export wizard, where the entire service definition is to be saved. Read more about the export function in Section 9.2 on page 204.	CTRL+E
Show Field	This property is only available in the context-sensitive menu, and marks the chosen field in the context field definition. This makes it easier to find and edit the field in the definitions.	CTRL+W
Delete	Deletes the selected element in the tree with all its contents.	DEL
Move up	Moves the selected element one step up in the tree.	CTRL+ARROW UP
Move down	Moves the selected element one step down in the tree.	CTRL+ARROW DOWN
Enter group	Moves the selected element into the group below.	CTRL+ARROW RIGHT
Leave group	Moves the selected element out of a group and places it above the group.	CTRL+ARROW LEFT

5.3.3.2 Context Fields

A context definition contains links to the elements defined under the **Context fields**. A link is created by dragging an element from the Context fields tree, and dropping it in the Context tree under a context node, see Figure 19.

The following elements can be linked as context fields.

- Data fields
- Top levels of hierarchical fields
- Map groups
- Array groups
- Collections
- Hierarchical field collections

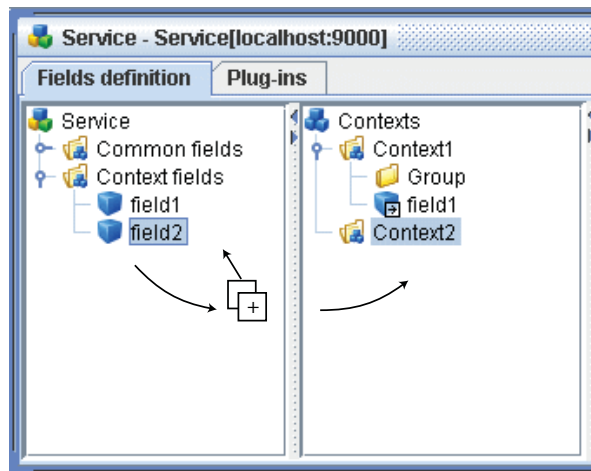


Figure 19 Creating a Link with Drag-and-Drop

5.3.4 Contexts

The Contexts tree can be edited by adding, editing and deleting elements. These elements are as follows:

- Context - The top node for a context, grouping field links.
- Field link - A link to a field defined under the Context fields node for a service.
- Group - Can be used for assembling related fields links. A group can contain both field links, and other groups. This grouping is also used when the fields are shown in the Simulation panel, see Figure 104.

Section 5.3.3.1 on page 37 describes how a context may be edited.

5.3.5 Editing a Field Group

A field group is used to assemble related data fields. The properties of a group element are shown in the configuration area when the element is selected.

The group properties are described in Table 23.

Table 23 Field Group Properties

Property	Description
Name	The name of the group.



Table 23 Field Group Properties

Property	Description
Type	<p>The type of this group. The options are:</p> <ul style="list-style-type: none">• Group• Array group• Map group <p>The different types of groups are further explained in Section 2.3.7 on page 8.</p> <p>(1)</p>
Key Fields	<p>This field is only visible when the group type is selected as Map group. It gives the identity of the field to be used as the key field for the value map group. This value is chosen from a list of the fields defined in the group.</p> <p>More than one field can be chosen. These fields are combined to create a composite key for the map group.</p>

(1) When the **Type** property of the group is changed, all the data fields in the group are removed.

An array group or map group is added to the service by adding a group in the fields definition and then changing the **Type** property on the group to **Array group** or **Map group**. Data fields created in an array or map group cannot be moved out of the group. Data fields created outside the array or map group cannot be moved into it.

When constructing a map group, one or several **Key fields** must be chosen for the group, from the fields defined in the group.

5.3.6 Editing a Data Field

By selecting a data field element in the tree, its properties are shown in the configuration area in the service editor. There are a number of properties that can be edited. Some of these properties are available depending on the data type of the field being defined.

The properties are described in Table 24.

Table 24 Properties for the Data Field

Property	Description
Name	The name of the data field, or hierarchical field. The case-sensitive name identifies the field and must be unique for data fields. For hierarchical fields, the name has to be unique at the same level, in a hierarchical structure.
Tag number	A numeric ID for the field, which may be used as a complement to the Name when identifying a field.



Table 24 Properties for the Data Field

Property	Description
Comment	An optional comment for the data field.
Data type	The data type for this data field. The data type is edited by choosing from a drop-down menu. More information about the available data types can be found in Section 5.3.12 on page 49.
Parameter type	Describes the origin of the data field. More information about the parameter type is be found in Section 5.3.6.1 on page 42.
Mandatory	Defines if the data field is mandatory. If a field is mandatory, it is selected by default in the simulator.
Show in plug-ins	<p>Determines if the data field should be available in standard ERE field-oriented plug-ins. If this check box is checked the data field is available, otherwise it is not.</p> <p>The availability of the field is also determined by the presence of constraints and the status of other settings such as parameter type and data type.</p>
Allow negative balance	For a field of the data type Amount , this check box determines if a negative balance input is allowed.
Amount Factory	For a field of the data type Amount , this parameter specifies a factory class for creating Amount objects. This feature is not currently supported in ERE.
Beginning of Time	<p>For a field of the data type Date, this parameter specifies a label for the Beginning of Time date constant.</p> <p>See Table 30.</p>
End of Time	<p>For a field of the data type Date, this parameter specifies a label for the End of Time date constant.</p> <p>See Table 30.</p>
Collection	<p>Defines if the field should be made a complex field. A collection may contain values, or data fields, identified by an index or key, instead of one single value. The options in this menu are:</p> <ul style="list-style-type: none"> • None • Array • Map • Set <p>The different types of collections are further explained in Section 2.3.6.1 on page 7.</p>



Table 24 Properties for the Data Field

Property	Description
Default	<p>The default value for this data field. It can be set to any value, regardless of the specified range for the field or any specified enumeration constant.</p> <p>If the collection parameter is set to array or map, an array editor is available here. A description of this is found in Section 6.7.1 on page 85.</p>
Input Restriction	<p>This field opens an editor for entering restrictions on the data which may be entered in the data field. The options are:</p> <ul style="list-style-type: none">• No Data• Enumeration• Range• Bit pattern• Relative date• Relative time <p>Section 5.3.6.2 on page 43 describes these restrictions in more detail.</p>
Editor class	<p>This field contains the fully qualified class name to an editor which takes care of the input to the field. This editor class must implement the Simulation Editor class, which may be found at <code>ericsson.ere.gui.interfaces.SimulationEditor</code>.</p> <p>This field is only visible if no input restriction has been chosen for the data field.</p>
Key	<p>This option is only visible in a hierarchical field collection, when the collection type is selected as MAP. It gives the identity of the field to be used as the key field for the map collection. This value is chosen from the data field defined in the collection by selecting the Key check box, see Figure 20.</p> <p>More than one data field can be chosen as a key field. These fields are combined to create a composite key for the map collection.</p>
Tooltip	<p>Fields can be defined with an optional description which is displayed in the selection tree editor, where a field is selected from a field chooser. The tooltip description is displayed when placing the mouse pointer over the target field name, see Figure 79. This makes it easier to know the purpose of the field.</p>

5.3.6.1 Parameter Type

The **Parameter type** defines how the field may be used in a tree. The six possible types are as follows:



- IN – The data field is read-only, and passes values into the selection tree for evaluation.
- OUT – The data field is read and write accessible, and is used to store the results of the tree execution. An out field has no initial value.
- VARIABLE – The data field can store values in the tree, for example, interim results of calculations.
- VALIDATION – The data field is used for test and debug purposes and cannot be included in the execution of a selection tree,
- VIRTUAL - The data field is read-only, and passes values into the selection tree for evaluation. The field is always mandatory in any service it is added to. The data in the field is not visible in the simulation panel.
- WORKING – This type is no longer used. It exists for backward compatibility reasons.

5.3.6.2 Input Restrictions Property

The **Input Restrictions** property affects how a value for a field is edited in the GUI and defines any restrictions on the values in the field. Input restrictions do not affect traffic in the ERE. The properties that are available for editing depend on the selected data type, but may include range, enumeration, bit pattern, relative dates and times and no data.

The different options are described in the following sub chapters.

5.3.6.2.1 Range

Some fields of numeric types have the option to change the **Range** of the input restrictions. This setting enables the user to set a minimum and maximum value for that specific field. The following data types are supported for range.

- UInt8
- UInt16
- UInt32
- Short
- Integer
- Long
- UShort
- UInteger
- ULong



— RatingDecimal

Both Min and Max fields have a check box next to them. Checking one of them sets the corresponding parameter to its minimum or maximum limit specified in the data type, and also disables it from being edited.

5.3.6.2.2 Enumeration

Fields of all data types have the option to set **Enumeration** in the input restrictions. This setting enables the user to define a set of values that can be used for this specific field. Choosing enumeration from the drop-down list reveals a parameter called enumeration and a button called **Table**. Selecting this button opens an array editor.

The array editor enables the user to set up a number of rows with name and value to define a value list.

The editor is explained further in Section 6.7.1 on page 85.

5.3.6.2.3 Bit Pattern

The input restrictions option **Bit pattern** is available for fields of data type Short, UShort, Integer, UInteger, Long, and ULong. This setting enables a bit pattern editor when editing the field in the GUI. Setting or unsetting the bits in the editor defines the data field. The editor is explained further in Section 6.7.7 on page 92.

5.3.6.2.4 Relative Date

Fields of the type Date have the option to set **Relative date** in the input restrictions. This allows an expression to be set up, defining a date to be relative to another specified date.

The editor is explained further in Section 6.7.10 on page 94.

5.3.6.2.5 Relative Time

Fields of the type Time have the option to set **Relative time** in the input restrictions. This allows an expression to be set up, defining a time to be relative to another specified time.

The editor is explained further in Section 6.7.11 on page 95.

5.3.6.2.6 No Data

Fields of all data types have the option to set **No data** in the simulation data. Choosing this option means that there are no specific editor settings for this field. This is the default setting for all fields.



5.3.7 Editing a Hierarchical Field

By selecting a hierarchical field in the tree, its properties are shown in the configuration area in the service editor.

The hierarchical field properties are described in Table 25.

Table 25 Hierarchical Field Collection Properties

Property	Description
Name	The name of the hierarchical field.
Collection	<p>The type of this collection.⁽¹⁾ The options are:</p> <ul style="list-style-type: none"> • NONE • ARRAY • MAP • SET <p>The different types of collections are further explained in Section 2.3.6.1 on page 7.</p>
Show in plug-ins	<p>Determines if the hierarchical field should be available in standard ERE field-oriented plug-ins. If this check box is checked the hierarchical field is available, otherwise it is not.</p> <p>The availability of the hierarchical field is also determined by the presence of constraints.</p>
Tooltip	Hierarchical fields can be defined with an optional description which is displayed in the selection tree editor, where a hierarchical field is selected from a field chooser. The tooltip description is displayed when placing the mouse pointer over the target hierarchical field name. This makes it easier to know the purpose of the hierarchical field.

(1) All fields in the hierarchical structure are removed if the **Collection** property of the top hierarchical field is changed.

When constructing a hierarchical field map collection, one or several key fields must be chosen for the collection, from the data fields defined in the collection. This is done by selecting the **Key** check box for the desired field, see the **Key** property in Table 24. In Figure 20, the data fields MSISDN and DoB are key fields. The key fields are symbolized with a key.

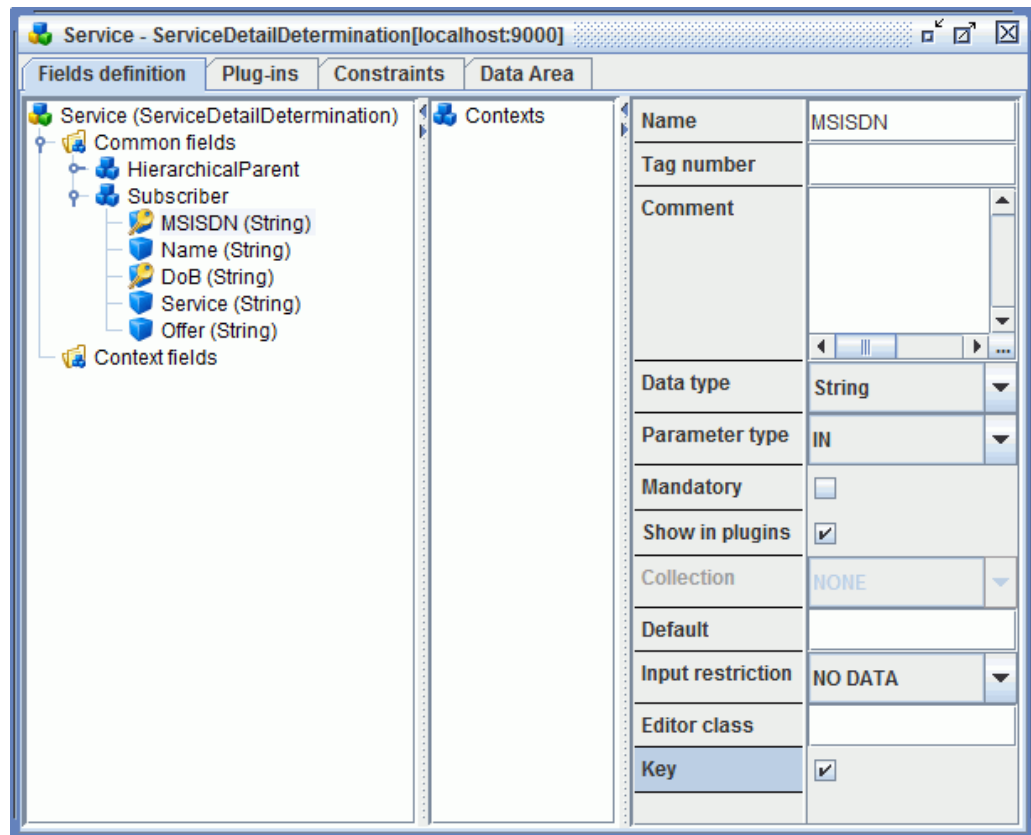


Figure 20 Hierarchical Map Key Field

5.3.8

Editing a Condition

By selecting a **Condition** element in the tree its settings are shown in the configuration area to the right, see Figure 21. There are two settings that can be edited.

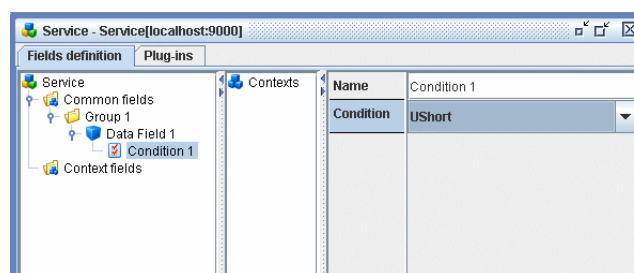


Figure 21 Editing a Condition

The properties are described in Table 26.



Table 26 Properties for the Field Condition

Property	Description
Name	The name of the condition. The name identifies the condition in the simulation panel.
Condition	The condition type that is to be used in this condition. This parameter is shown as a drop-down menu, containing all available conditions that matches the data type of the connected data field.

5.3.9 Editing the Context Root Element

By selecting the **root** node in the Context tree its property is shown in the configuration area to the right, see Figure 22. There is one property that can be edited.

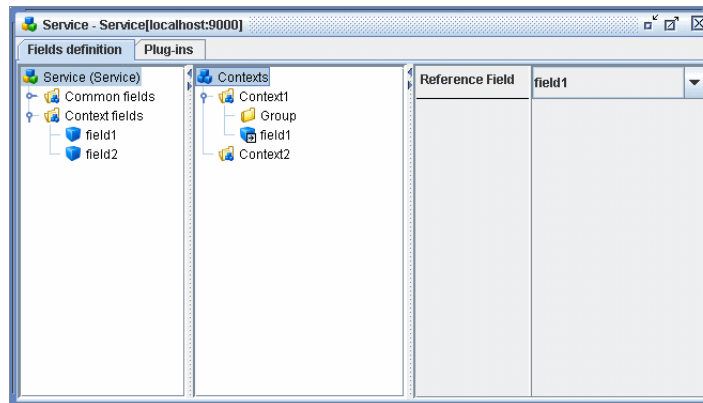


Figure 22 Editing the Context Root Element

The property is described in Table 27.

Table 27 Properties for the Context Root Element

Property	Description
Reference field	Chosen from the defined context data fields. This data field is used to define which Context that should be used.

5.3.10 Editing a Context Element

By selecting a **Context** element in the Context tree its properties are shown in the configuration area to the right, see Figure 23. There is one property that can be edited.

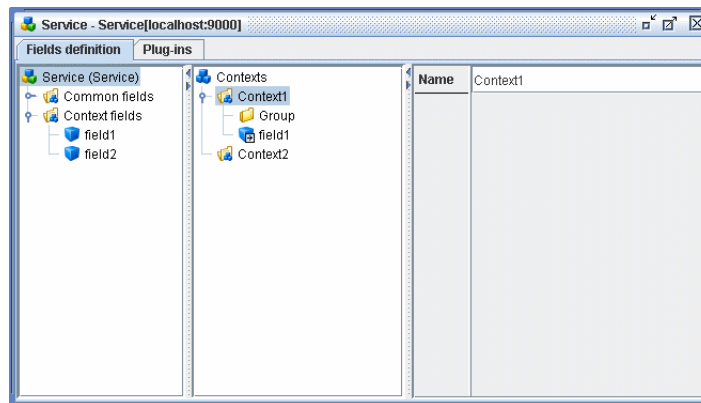


Figure 23 Edit a Context Node

The property is described in Table 28.

Table 28 Properties for a Context Element

Property	Description
Name	The name of the context. The name identifies the context in the simulation panel.

5.3.11 Editing a Context Group Element

By selecting a **Group** element in the tree its properties are shown in the configuration area to the right. The Group properties are shown as in Figure 24. There is one property that can be edited.

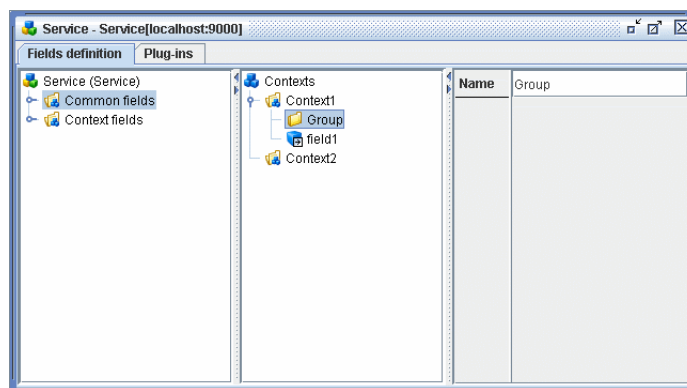


Figure 24 Editing a Context Group

The property is described in Table 29.

Table 29 Context Group Properties

Property	Description
Name	The name of the group.



5.3.12 Data Types

The data types available in ERE are for the most part standard data types, although some have been developed specially for the application.

The available data types are shown in Table 30.

Some of the following data types are being phased out, and are marked as such if chosen in the GUI.

Table 30 Available Data Types

Name	Description	Range/Format
Amount	Decimal value and currency code for input of a monetary amount.	
BcdString	Binary Coded Decimal String values. This string is a Telephony Binary Coded Decimal (TBCD). The TBCD string is different from the ordinary Binary Coded Decimal (BCD) string in the way that it allows not only 0-9 but also A-F. The F however, is only used as a filler and is removed from the editor.	0-9, A-F
Boolean	Java Boolean	True or false
Date	Java Date The Date data type is also used to set the constants Beginning of Time and End of Time . The constants define dates furthest in the past and in the future, respectively, that are available in the application. They may, for example, be set to 'Always On' and 'Never Expires'.	YYYY-MM-DD
Decimal	Decimal value. The Decimal data type gives control of the decimal scale, that is the original scale is used after calculations performed.	Min value: -1.7976931348 623157e308 Max value: 1.7976931348 623157e308
Double	Java Double	Min value: -1.7976931348 623157e308 Max value: 1.7976931348 623157e308
Integer	Java Integer	Min value: -2147483648 Max value: 2147483647



Table 30 Available Data Types

Name	Description	Range/Format
Long	Java Long	Min value: -9223372036854775808 Max value: 9223372036854775807
MonetaryUnits	Decimal value for money. The MonetaryUnit data type gives control of the decimal scale, that is the original scale is used after calculations has been performed.	Min value: -1.7976931348623157e308 Max value: 1.7976931348623157e308
Object	An data type which is not pre-defined. See Section 5.3.12.1 on page 50 for more details.	
OctetString	OctetString holds an array of bytes for the data.	0-9, A-F
RatingDecimal	Decimal value for monetary amount.	
Short	Java Short	Min value: -32768 Max value: 32767
String	Java String	No limit defined
Time	Date and time variable	YYYY-MM-DD hh:mm:ss
UInteger	Unsigned Integer	Min value: 0 Max value: 4294967295
ULong	Unsigned Long	Min value: 0 Max value: 18446744073709551615
UnsignedInt8	8 bit Unsigned integer value	Min value: 0 Max value: 255
UnsignedInt16	16 bit Unsigned integer value	Min value: 0 Max value: 65535
UnsignedInt32	32 bit Unsigned integer value.	Min value: 0 Max value: 4294967295
UShort	Unsigned Short	Min value: 0 Max value: 65535

5.3.12.1 Object

This is a type which may be defined.



Fields of the type object must define the following:

- An **Editor Class**. This contains the fully qualified path to an editor which takes care of the input to the field. More information on the editor class is available in Table 24.
- A **Value Class**. This contains the fully qualified path to a class that defines the Java type associated with the field.
- A **Value Class Factory**. The defined value factory must implement the ValueClassFactory interface. The value class factory interface may be found at `ericsson.ere.datatype.interfaces.ValueClassFactory`.

5.4 Plug-In Editor

When choosing the **Plug-ins** tab in the **Service Editor**, the window is shown with a tree structure on the left and a configuration area on the right, see Figure 25. The configuration area on the right shows the properties for the element currently selected in the tree.

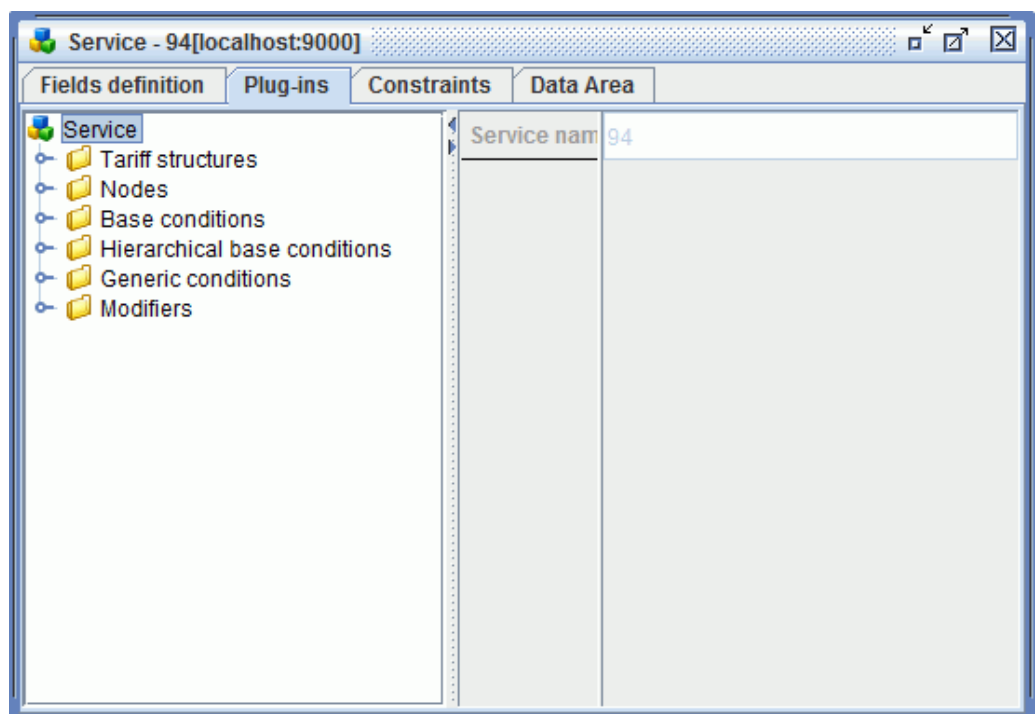


Figure 25 The Plug-In Declaration Editor

The tree on the left contains the installed plug-ins, grouped in folders. The groups are as follows:

- Tariff structures - the root of a tree. This is the starting point for the selection or decision execution.

- Nodes - groups conditions, nodes, and modifiers. If the conditions under a node is evaluated true, the node opens and the execution continues. This mechanism makes it possible to take different paths and decisions in the tree depending on the value of the fields.
- Base Conditions - conditions that can be connected to a data field. This condition plug-in is data type aware, which makes it possible to use the condition towards any field of the supported data type. Base condition plug-ins are mapped to fields in the fields editor.
- Hierarchical base conditions - Hierarchical base conditions are conditions used in a selection tree, where they can be assigned to a specific hierarchical field using the FieldSelection condition. A hierarchical base condition can be assigned to any hierarchical field with matching collection type. Which collection types a hierarchical base condition supports is specified in the service.
- Generic Conditions - conditions that know how to get the value to be evaluated. These conditions do not need any additional information. Normally, they are service specific.
- Modifiers - the point in the selection tree where the data is manipulated, for example, a field value can be set, calculating a price, and so on.

The Plug-in declaration can be edited by adding, editing or deleting plug-ins in the groups. The groups themselves cannot be edited or deleted and new groups cannot be added.

Editing in the tree can be done by using the context-sensitive menu which is shown when right - click a tree element or by using the keyboard. The actions that can be taken in the tree are described in the next sub chapter. When saving the definition or selecting another tab in the Service Editor, the Plug-ins definition is validated and any inconsistent inputs are reported in a popup message.

5.4.1 Editing the Plug-In Declaration

The plug-in declaration can be edited using the context-sensitive menu or by pressing the key combination connected to an action. The menu is accessed by right - click anywhere in a tree structure. The menu is context-sensitive and offers different editing alternatives depending on what kind of tree element that was selected, see Figure 26. It offers different editing alternatives depending on what kind of tree element that was selected.

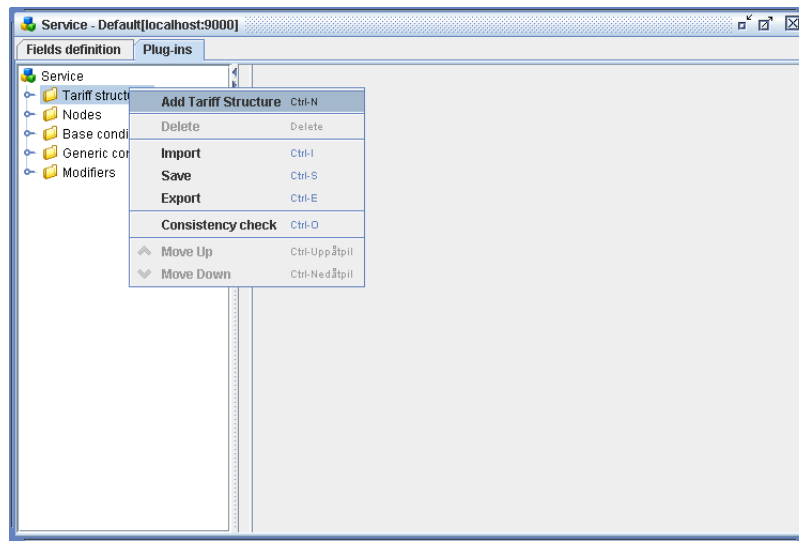


Figure 26 Context-Sensitive Menu

5.4.1.1 Context-Sensitive Menu

Table 31 explains which actions that are available and the corresponding key combination.

All alternatives are not applicable for every element type.

Table 31 Plug-In Menu Alternatives

Action	Description	Hot Key
Add Node	Adds a new node plug-in. Only shown when the Nodes folder or an existing node is selected. Read more about the Node in Section 5.4.3 on page 55.	CTRL+N
Add Base Condition	Adds a new base condition plug-in. Only shown when the Base conditions definition folder or an existing base condition is selected. Read more about the Base Conditions in Section 5.4.4 on page 55.	CTRL+N
Add Generic Condition	Adds a new generic condition plug-in. Only shown when the Generic conditions definition folder or an existing generic condition is selected. Read more about the Generic Conditions in Section 5.4.6 on page 59.	CTRL+N
Add Modifier	Adds a new modifier plug-in. Only shown when the Modifiers folder or an existing modifier is selected. Read more about the Modifiers in Section 5.4.7 on page 60.	CTRL+N
Delete	Deletes the selected plug-in.	DEL
Import	Opens an import wizard where a file to import can be chosen. The file must contain XML in importable format. Read more about the import function in Section 9.1 on page 203.	CTRL+I

Table 31 Plug-In Menu Alternatives

Action	Description	Hot Key
Save	Saves the current plug-in declaration.	CTRL+S
Export	Opens a file directory dialog window where a file to export to can be chosen. The whole Service Definition is exported. Read more about the export function in Section 9.2 on page 204.	CTRL+E
Move up	Moves the selected element one step up in the folder.	CTRL+ARROW UP
Move down	Moves the selected element one step down in the folder.	CTRL+ARROW DOWN

Note: If a plug-in is used in the Field definition tab, its declaration is locked and cannot be changed.

5.4.2 Editing a Tariff Structure

By selecting a **Tariff Structure** element in the tree, its properties are shown in the configuration area to the right, see Figure 27. There are a number of properties that can be edited.

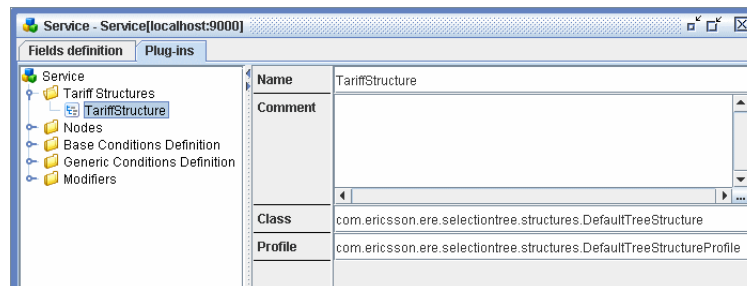


Figure 27 Editing a Tariff Structure

The properties are described in Table 32.

Table 32 Properties for the Tariff Structure Element

Property	Description
Name	The name of the tariff structure.
Comment	An optional comment for the tariff structure.
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.



5.4.3 Editing a Node

By selecting a **Node** element in the tree, its properties are shown in the configuration area to the right, see Figure 28. There are a number of properties that can be edited.

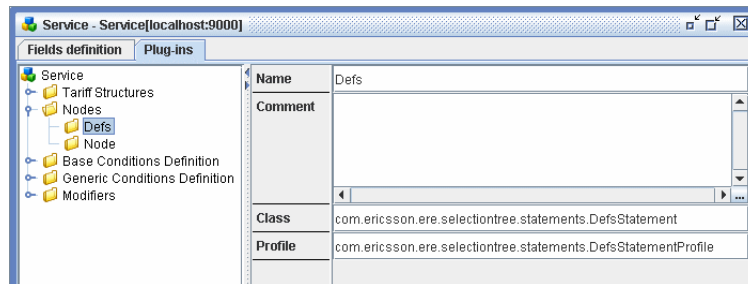


Figure 28 Editing a Node

The properties are described in Table 33.

Table 33 Properties for the Node

Property	Description
Name	The name of the node.
Comment	An optional comment for the node.
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.

5.4.4 Editing a Base Condition

By selecting a **Base Condition** element in the tree its properties are shown in the configuration area to the right, see Figure 29. There are a number of properties that can be edited.

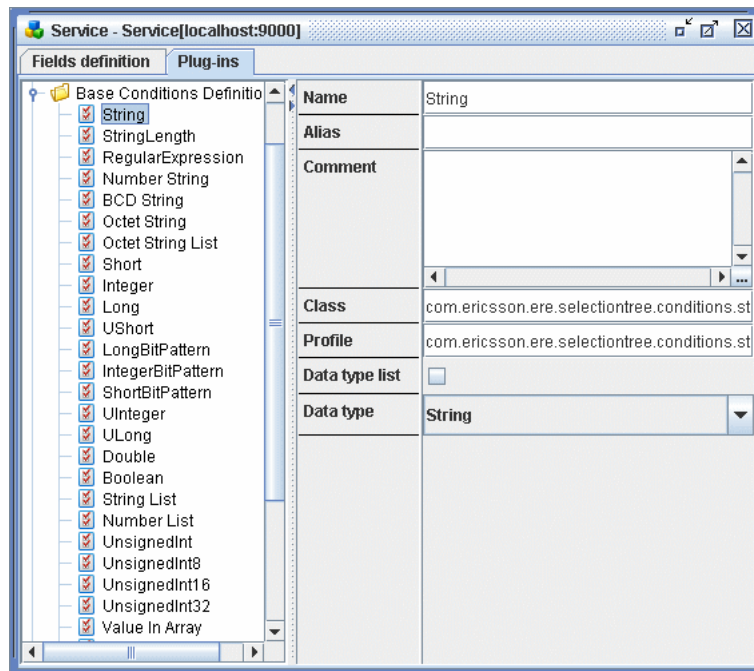


Figure 29 Editing a Base Condition

Note: Creation or modifications of a service is only to be performed by qualified personnel.

The Plug-in declaration editor icons are described in Table 34.

Table 34 Description of Base Condition Icons

Condition/Icon	Description
	Base condition icon.
	Locked base condition icon.

The properties are described in Table 35.

Table 35 Properties for the Base Condition

Property	Description
Name	The name of the base condition. This name is used in settings.
Alias	An optional name for the plug-in shown to the user in the GUI.
Comment	An optional comment for the base condition.
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.



Table 35 Properties for the Base Condition

Property	Description
Data type list	Defines if the condition should be able to handle a single or multiple data types. If this check box is checked the data type parameter is treated like an array and enables defining multiple data types. The Data type list is described further in Section 5.4.4.1 on page 57.
Data type	The data type that this condition should handle. Only data fields of this defined data type are allowed to use this condition. The possible data types are described in Section 5.3.12 on page 49. If the parameter Data type list is set this property is replaced by a Table button. The Data type list property is described further in Section 5.4.4.1 on page 57.

5.4.4.1 Data Type List Parameter

By setting the property **Data type** list, the base condition can be used by data fields of all data types defined in the list instead of a single one. When the Data type list property is set, the Data type parameter is replaced by a **Table** button which enables an array editor. The editor allows the user to define a list of data types, see Figure 30.

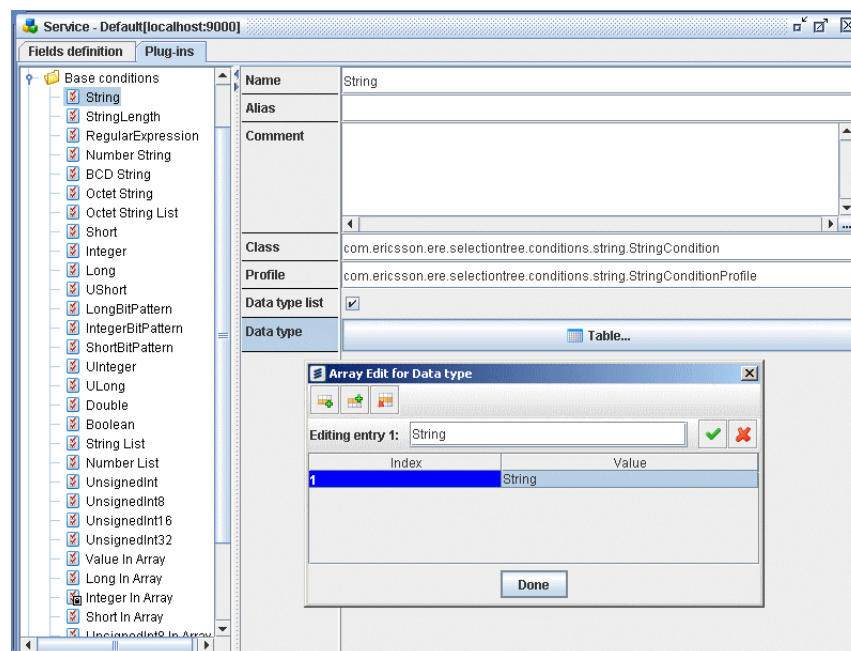


Figure 30 Data Type List Editor

The array editor is explained further in Section 6.7.1 on page 85.

5.4.5 Editing a Hierarchical Base Condition

By selecting a **Hierarchical base condition** element in the tree its properties are shown in the configuration area to the right, see Figure 31. There are a number of properties that can be edited.

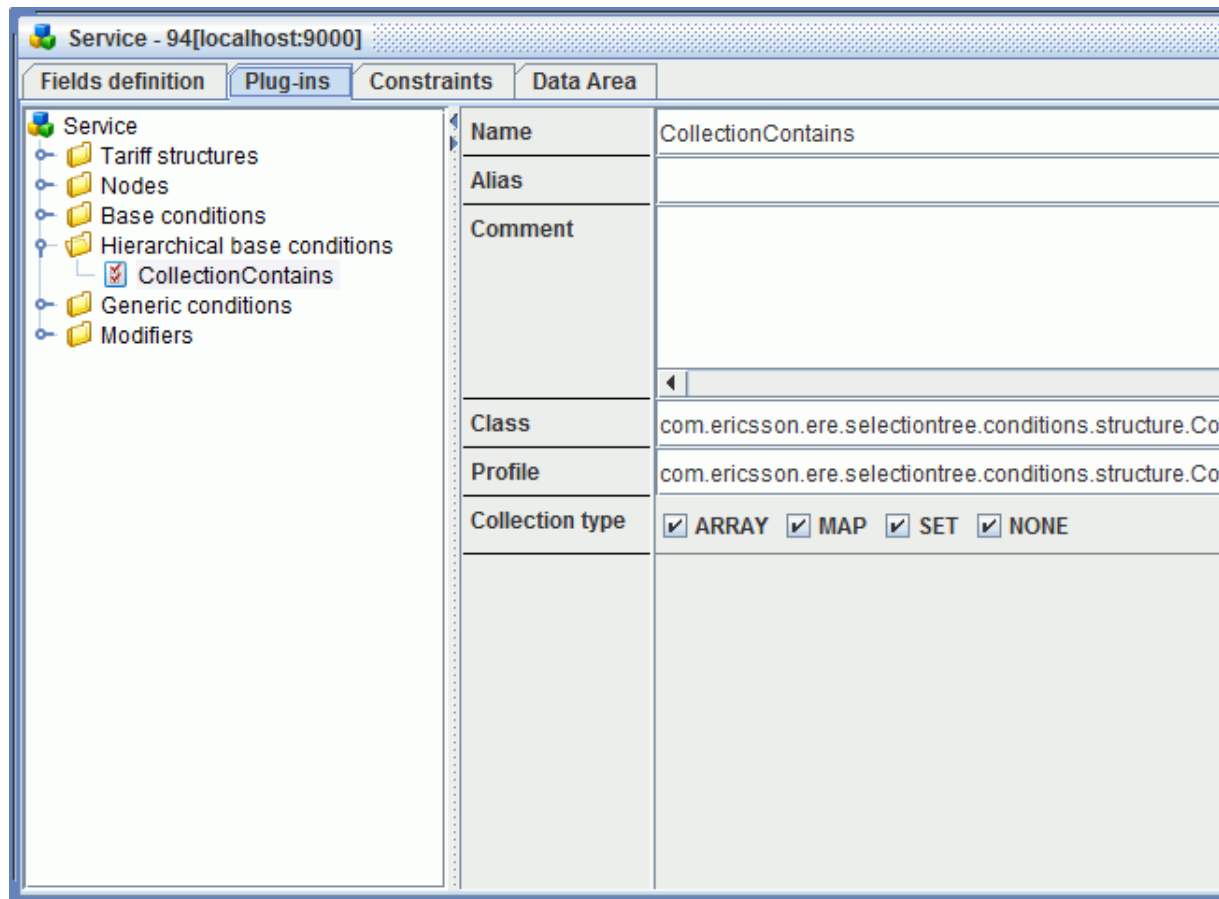


Figure 31 Editing a Hierarchical Base Condition

The properties are described in Table 36.

Table 36 Properties for the Hierarchical Base Condition

Property	Description
Name	The name of the hierarchical base condition. This name is used in settings.
Alias	An optional name for the plug-in shown to the user in the GUI.
Comment	An optional comment for the hierarchical base condition.



Property	Description
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.
Collection type	Defines what collection type the hierarchical base condition can operate on.

5.4.6 Editing a Generic Condition

By selecting a **Generic Condition** element in the tree, its properties are shown in the configuration area to the right, see Figure 32. There are a number of properties that can be edited.

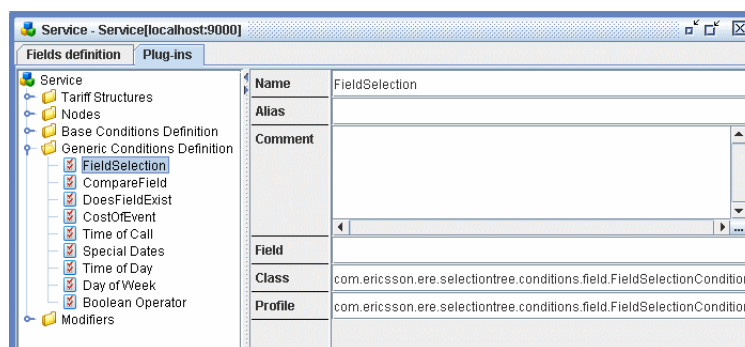


Figure 32 Editing a Generic Condition

The parameters are described in Table 37.

Table 37 Properties for the Generic Condition

Property	Description
Name	The name of the generic condition.
Alias	An optional name for the plug-in shown to the user in the GUI.
Comment	The path to the profile plug-in taking care of the user input from the GUI.
Field	Name of a field that can be used to give additional information to the plug-in.
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.

5.4.7 Editing a Modifier

By selecting a **Modifier** element in the tree, its properties are shown in the configuration area to the right, see Figure 33. There are a number of properties that can be edited.

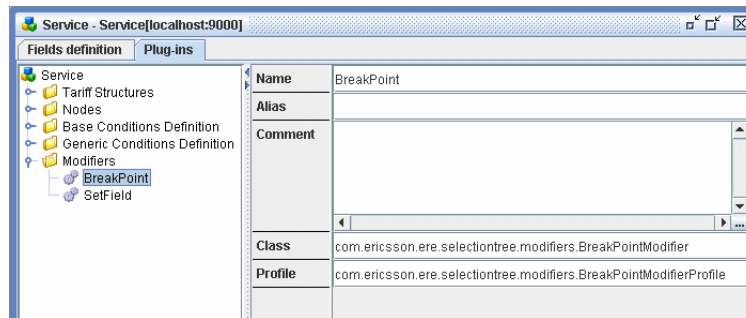


Figure 33 Editing a Modifier

The properties are described below in Table 38.

Table 38 Properties for the Modifier

Property	Description
Name	The name of the modifier.
Alias	An optional name for the plug-in shown to the user in the GUI.
Comment	An optional comment for the modifier.
Class	The path to the class plug-in that is used in traffic execution.
Profile	The path to the profile plug-in taking care of the user input from the GUI.

5.5 Constraints

The **Constraints** tab in the **Service Editor** makes it possible to include, exclude, or select elements from the GUI. Such as defined fields, conditions, modifiers, and operations. The constraints defined here are only applied to the objects in the GUI, and are not applied to the traffic cases handled by the engine. Figure 34 shows an example of the constraints tab.

Field constraints override the **Show in plugins** property in the data field definition, see Table 24 for more details on this function.

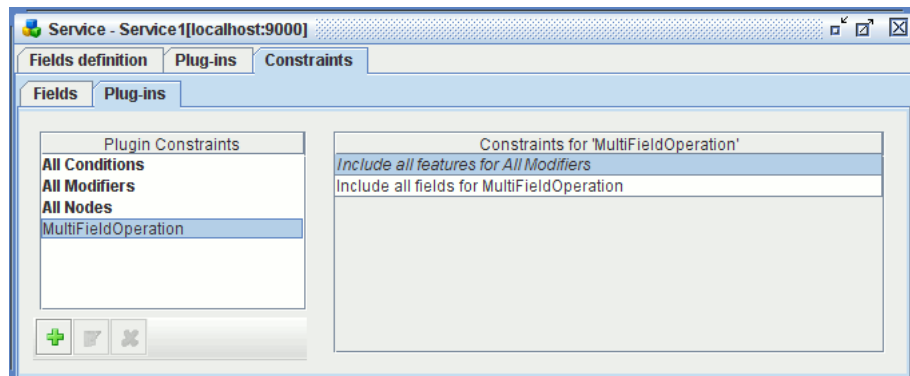


Figure 34 Constraints Tab in Service Editor

The **Constraints** tab has two further tabs, **Fields** and **Plug-ins**. Selecting either of these tabs allows constraints to be added, edited or deleted. The dialog which opens to allow editing of the constraint is different in both cases. The parameters in the dialog depend on what is chosen in the **Plugin to constrain** field. Figure 35 is an example of a plug-in constraint dialog.

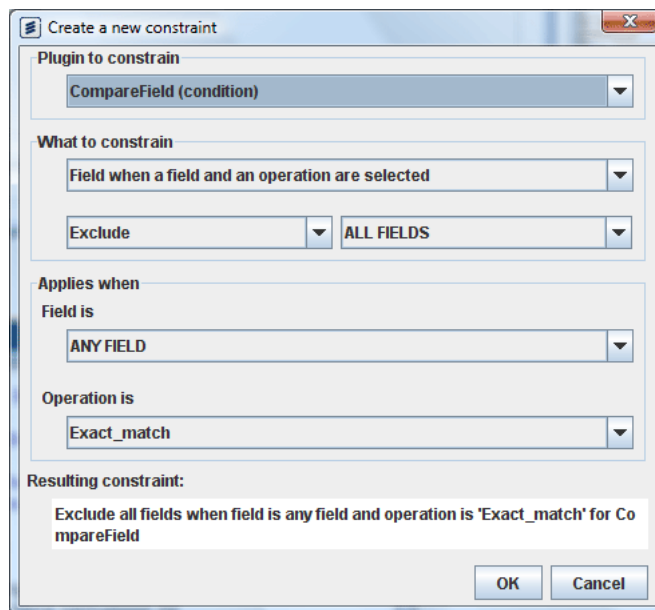


Figure 35 An Example of a Constraints Dialog

Table 39 Properties in a Constraints Dialog

Property	Description
Plugin to constrain	The name of the plug-in (condition, modifier or node) to constrain. All conditions or all modifiers may also be chosen.



Table 39 Properties in a Constraints Dialog

Property	Description
What to constrain	<p>This parameter allows the definition of which property of the plug-in is to be constrained. It is dynamic, depending on the plug-in type, and may include:</p> <ul style="list-style-type: none">• Condition• Feature• Field• Operation <p>Selection of how to constrain the selected property is also dynamic and may include:</p> <ul style="list-style-type: none">• Include - Target to include according to the selected plug-in.• Exclude - Target to exclude according to the selected plug-in.• Select - Target to select as a default condition. Select is only available for the FieldSelection condition, when Condition when a field is selected is chosen in What to constrain. <p>The target to constrain may be any available field, operation, or feature, for the selected plug-in.</p>
Applies when	<p>It is a condition to fulfill the constraint, in conjunction with a specific field, or field and operation. The parameter is dynamic and may include:</p> <ul style="list-style-type: none">• Fields is• Operation is <p>This property is only available for modifier constraints, when either of the following is selected in What to constrain.</p> <ul style="list-style-type: none">• Field when a field is selected• Operation when a field is selected• Field when a field and an operation are selected <p>The data type of the selected field affects the choices available in the drop-down menus.</p>
Resulting constraint	The configured constraint described in words.

Highlighting a constrained target in the list under the constraints tab gives a summary of all the constraints which have been defined for that target. In the example in Figure 34, the MultiFieldOperation modifier has two constraints defined. This list is ordered so that the most specific constraints are always at



the bottom. This makes it possible to define more general constraints for many targets, and even more specific constraints for some targets.

5.5.1 Field Constraints

Field constraints make it possible to constrain fields when working with conditions and modifiers. Under the **Fields** tab, when selecting the **Add new constraint** or **Edit settings for constraint** buttons, a dialog opens in which the constraint may be defined.

It is possible to add more than one constraint simultaneously for the same condition or modifier. For example, it is possible to constrain all conditions, and define further constraints for an individual condition.

5.5.2 Plug-in Constraints

Plug-in constraints make it possible to specify constraints when working with a selected plug-in. Under the **Plug-ins** tab, when selecting **Add new constraint** or **Edit settings for constraint** buttons, a dialog opens in which the constraint may be defined.

5.5.2.1 Feature Constraint

Under the **Plug-ins** tab, it is possible to constrain some features.

Only certain plug-ins allow features to be constrained, and selecting a plug-in of this type will allow **Feature** to be selected in the **What to constrain** parameter. Features may then be included or excluded. By default, features are disabled.

5.6 Service Data Area

In the **Data Area** tab of the **Service Editor**, it is possible to define data elements that are available for the current service.

Fields definition		Plug-ins		Constraints		Data Area	
Data Elements				Name	SampleSingleValue		
SampleSingleValue				Element type	Single value		
SampleXMLFragment				Value type	Integer		
				Value	23		
<div> <div>+</div> Add <div>×</div> Delete </div>				<div> <div>✓</div> Commit <div>×</div> Cancel </div>			

Figure 36 An Example of Service Data Area

The parameters that can be set in the data area are described in Table 40.



Table 40 Properties for the Data Area

Property	Description	Comment/Available Fields
Name	The name of the data element.	The name must be unique within the service.
Element Type	Single value - A single element is defined, of any of the data types supported by ERE.	<ul style="list-style-type: none">• Value Type - A drop-down menu from which the data type of the element is chosen. The available value types are the same as those shown in Table 30.• Value Class Factory - If a data type of the type Object is chosen, the path to the factory that defines the value classes is entered here.• Value Class - If a data type of the type Object is chosen, the path to a class that defines the Java type associated with the value is entered here.• Value - The value of the data element. Only values that agree with data type chosen in Value Type can be entered. If the value type is an object, the Value is a string that represents a value of the type in the Value Class.
	Array - An array is defined, where the values in the array can be of any of the data types supported by ERE.	Array Values - An array editor, in which the array of values for the element is entered. Only values that agree with data type chosen in Value Type can be entered.
	Map - A map of values is defined, where the key type and map values can be of any of the data types supported by ERE.	<ul style="list-style-type: none">• Key Type - A drop-down menu from which the data type of the map key is chosen.• Map Values - A map editor, in which the map of values for the element is entered. Only keys and data values that agree with data type chosen in Key Type and Value Type can be entered.
	XML fragment - An XML fragment is defined.	XML fragment - The fragment must contain one, and only one, root node.

A new data element is added to the service by using the **Add** button. A data element is removed from the service by using the **Delete** button.

An existing data element can be edited by selecting the element from the **Data Element** list. If a data element is entered with an invalid format or incomplete information, a pop-up appears with an error message when **Commit** is chosen.



The changes are discarded and the old values retained when the **Cancel** button is chosen.

The changes are first implemented in the service when the service is saved.

5.7 Saving the Service Definition

The configurations have to be saved before closing the Service Definition editor. This can be done in the following ways:

- Choose **Save** from the context-sensitive menu.
- Click **Save** in the RMA main toolbar, with the **Service Definition** editor window selected.
- Press **CTRL+S**, with the **Service Definition** editor window selected.

If the editor is closed with unsaved changes the user is asked if the configuration should be saved before it is closed, see Figure 37.

When choosing **Cancel** nothing happens and editing is still possible.

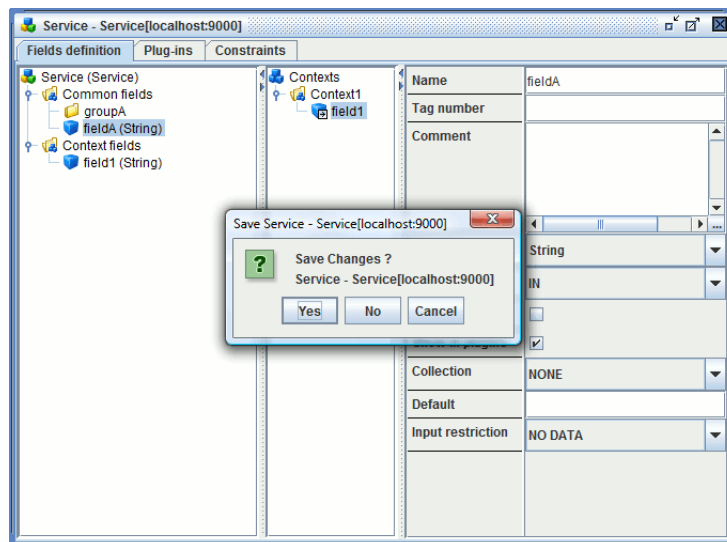


Figure 37 Save Changes Dialog

The Service Definition cannot be saved if any of the configured data is invalid. When the user tries to save an invalid configuration a notification is shown.

Changing a Service Definition can affect existing rating periods that are based on it. When the user tries to save a Service Definition that is in use, the editor warns that improper changes can make existing rating periods fail. It is possible to choose to **Cancel** the save or **Force the upgrade**, see Figure 38.



Figure 38 Force Upgrade Dialog

Existing rating periods using a service definition may be affected negatively by changes made to the service definition. For instance, if a condition or modifier plug-in used in the selection tree has been removed from the service definition or if a map group has no assigned key field, the tree is invalid (see Figure 39).

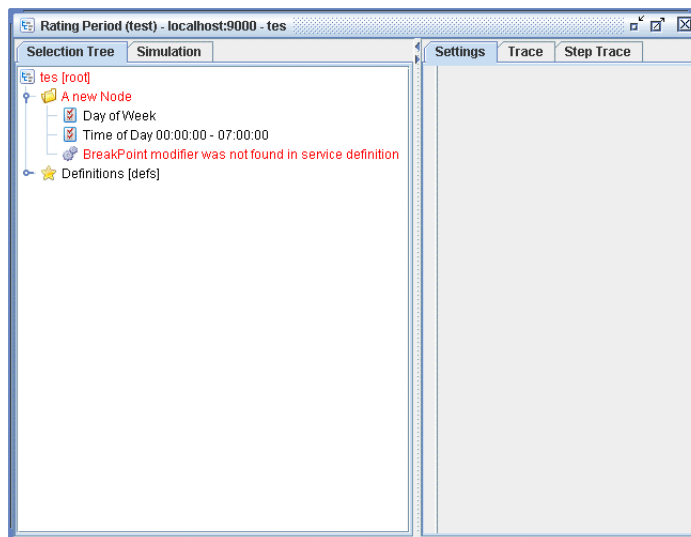


Figure 39 Missing Plug-in



6 Working with Rating Periods

The selection tree editor is a tool for configuring Selection Trees in ERE. It can be accessed from the Rating Manager by choosing a **Rating Period** in **Service Provider List** and double-clicking it or by right-clicking on the desired **Rating Period** and selecting **Edit Selection Tree**.

The actions in the Selection Tree Editor include the following:

- Adding new Selection Trees, see Section 6.1 on page 68
- Editing Selection Trees, see Section 6.2 on page 71
- Setting up the logic in the Selection Tree with the different elements, see Section 6.3 on page 71.
- Saving Selection Trees, see Section 6.12 on page 109.
- Import and Export of a Selection Tree, see Section 9.1 on page 203 and Section 9.2 on page 204.
- Adding Bookmarks, to bookmark certain elements in a Selection Tree, see Section 6.13 on page 109.
- Using the find function for locating certain elements in the Selection Tree, see Section 9.5 on page 211.

The Selection Tree Editor consists of two areas, see Figure 40:

The Selection Tree tab is located in the left part of the Selection Tree Editor and it is where the Selection Tree connected to the selected Rating Period is set up with its logic.

The Settings tab is located in the right part of the Selection Tree Editor and is used for the specific settings for each item in the Selection Tree.

Settings are shown for the tree element that is selected (marked) in the Selection Tree. An important thing to mention is the possibility to simulate an execution of the logic in the tree, which is done in the Simulation tab, along with the Trace and Step Trace tabs. This is further explained in Section 7 on page 113.

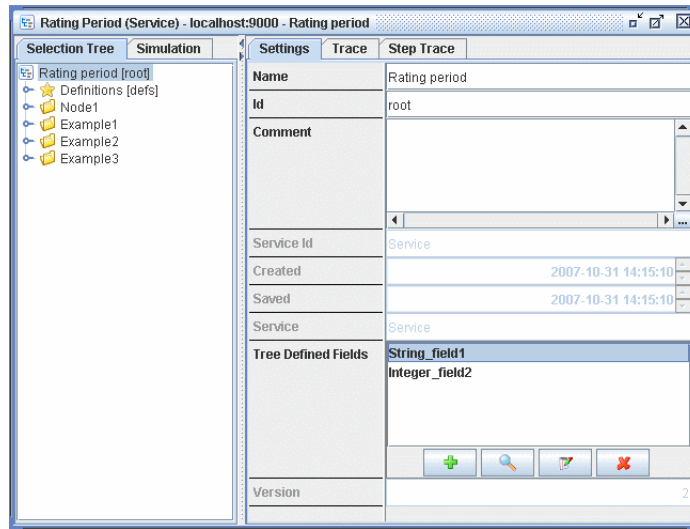


Figure 40 The Two Tabs

6.1 Creating a New Rating Period

Creating a new Rating Period is done in several steps. The first one is to create the Rating Period for the Selection Tree, which holds the start date for execution of the tree, and to specify what Service it should be connected to. The Service, consisting of all building blocks possible to use in the tree must also be specified along with the container for Rating Periods with the same Service, called Rating Plan.

A new Rating Period is created by a right - click on an element in the ERE-navigator tree (Service Provider List) on the level where the user wants to position the Selection Tree. It is possible to do so as high as the Service Provider list folder. Depending on what level in the tree that is clicked, different parts of the New Rating Period dialog are automatically filled in advance. The properties of the New Rating Period dialog are found in Table 41. An example of adding a New Rating Period dialog, when selecting Add a New Rating Period as high as the Service Provider List, see Figure 41, Figure 42 and in a Rating Plan Figure 43 and Figure 44.

It is possible to sort the rating Periods either by name or by date. This is done from **Settings** in **File** menu.

The options that can be chosen are described in Table 42.

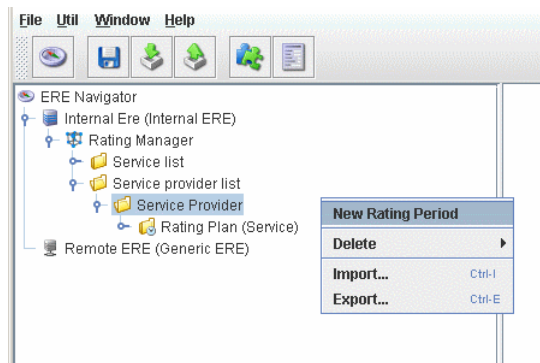


Figure 41 Add a New Rating Period as High as the Service Provider List

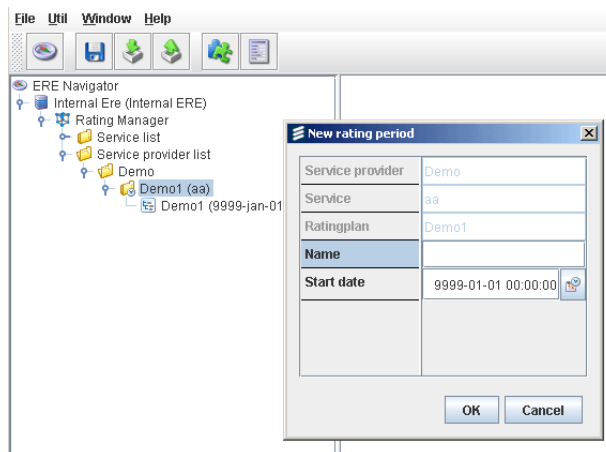


Figure 42 Add a New Rating Period as High as the Service Provider List, Continue

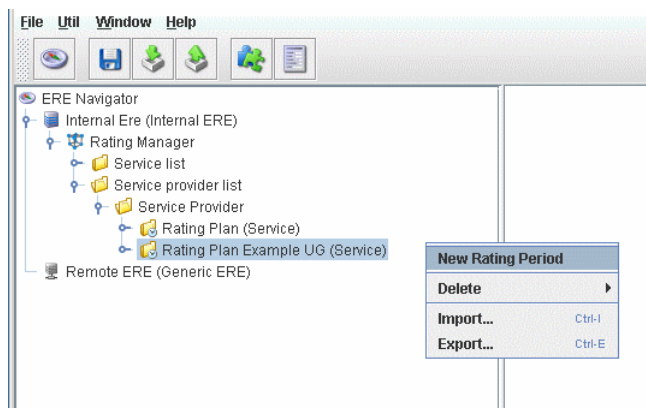


Figure 43 Add a New Rating Period in a Rating Plan.

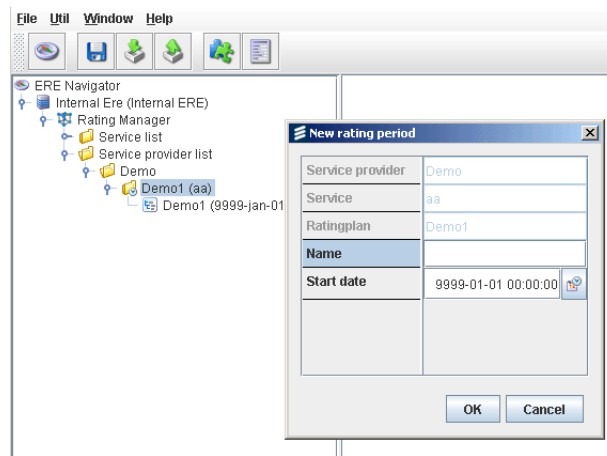


Figure 44 Add a New Rating Period in a Rating Plan, Continue

Table 41 Properties for the New Rating Period Dialog

Property	Description
Service Provider	<p>Specify what Service Provider the Selection Tree should be grouped under. It is possible to enter the name of an existing Service Provider. If a new name is specified, a Service Provider with the given name is created.</p> <p>This field may be filled in advance if the New Rating Period is done on a Service Provider or lower.</p>
Service	<p>Specify what service the Selection Tree should be connected to. All Services available in the Service List is found in the drop-down. To create a new Service, see section Section 5.1 on page 33.</p> <p>This field may be filled in advance if the New Rating Period is done on a Service Provider or lower.</p>
Rating plan	<p>Specify what Rating Plan this Selection Tree should be contained in. Either enter a new name to create a new Rating Plan, or enter a name of an existing Rating plan. Rating Periods with the same Service can be contained in the same Rating Plan. All Rating Periods that are contained in the Rating Plan must have different start dates.</p> <p>This field may be filled in advance if the New Rating Period is done on a Rating Plan.</p>
Name	Enter the name of the Rating Period to be created.
Start date	Enter the start date of the Rating Period.
Rating rule tree file	Not possible to edit. If the Rating Period has been saved, the file name is shown here if Properties has been chosen on an existing Rating Period.

Table 42 Options in the New Rating Period Dialog

Option	Description
OK	Completes the creation and closes the dialog.
Cancel	Cancels the creation.



6.2 Opening a Selection Tree for Editing

All created Rating Periods are opened for viewing and editing by double-clicking on the wanted Rating Period in the ERE-Navigation Tree. It is also possible to right - click on a **Rating Period** and choose **Edit Selection Tree** in the context-sensitive menu. The editor is displayed as in Figure 45.

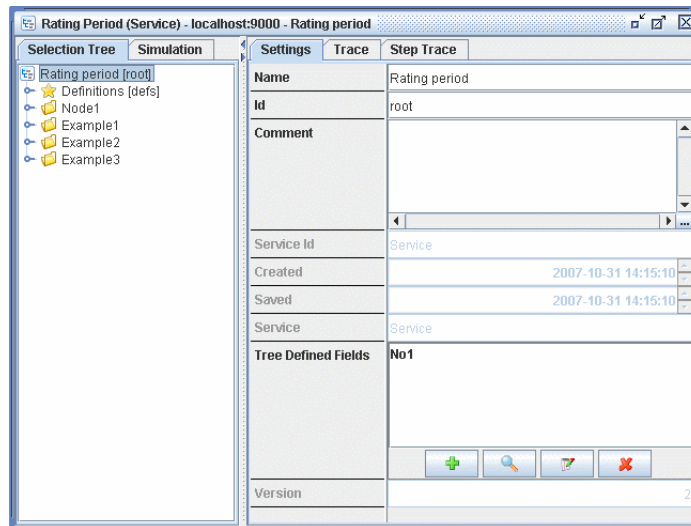


Figure 45 Editing a Selection Tree

The Selection Tree Editor opens with the selected Selection Tree in the application area of RMA.

It is possible to open several instances of one Selection Tree or to have different Selection Trees open at the same time. This makes it possible to, for example, compare two trees or to copy parts from one tree to another.

6.3 Selection Tree Control Commands

The Selection Tree Editor context-sensitive menu contains all possible commands of the items in the selection tree. The menu is accessed by right - click in the Selection Tree. The menu is context-sensitive, which means that some menu alternatives that are not applicable for the current tree element in the selected tree may be disabled.

Drag - and -drop and undo or redo commands can be used for elements in the Selection Tree, as shown in Section 6.3.3 on page 76.

6.3.1 Selection Tree Menu Options

This section describes the menu options in the Selection Tree menu, see Figure 46.

The descriptions are presented in Table 43.

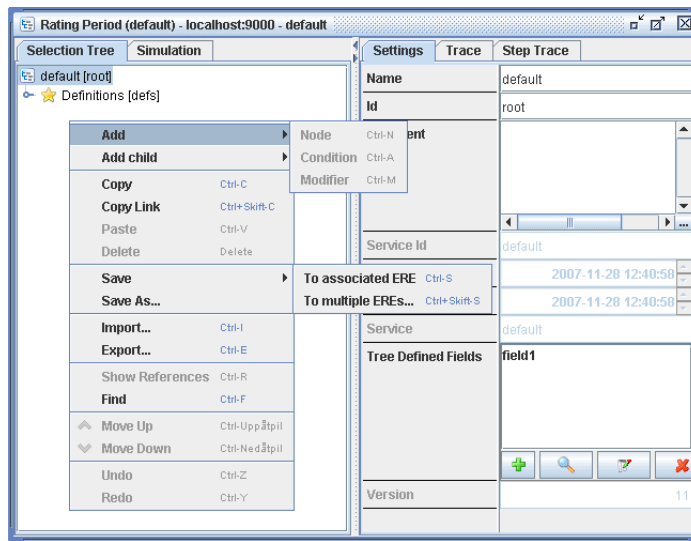


Figure 46 Selection Tree Menu

Table 43 The Selection Tree Menu Options and Their Descriptions

Menu option	Description	Hot key
Add	Node - Adds a new node after the selected node or condition. Enabled everywhere except if the insertion point is the root of the tree. The node is inserted below the insertion point, on the same level.	Ctrl + N
	Condition - Adds a new condition in the selected node. Enabled if the insertion point is a condition or modifier on the same level. The condition is inserted below the insertion point, on the same level.	Ctrl + A
	Modifier - Adds a new modifier to the selected node. Enabled if the insertion point is a condition or modifier on the same level. The modifier is inserted below the insertion point, on the same level.	Ctrl + M



Table 43 The Selection Tree Menu Options and Their Descriptions

Menu option	Description	Hot key
Add child	Node - Adds a sub-node in the selected node. Enabled if the insertion point is another node, the root or Definitions. The node is inserted below the insertion point, on a sublevel.	Ctrl + Shift + N
	Condition - Adds a new condition in a sublevel to the selected insertion point. Enabled if the insertion point is Definitions or a node. The condition is inserted below the insertion point, on a sublevel.	Ctrl + Shift + A
	Modifier - Adds a new modifier in a sublevel to the insertion point. Enabled if the insertion point is Definitions or a node. The modifier is inserted below the insertion point, on a sublevel.	Ctrl + Shift + N
Copy	Copies the selected part of the tree to the copy buffer without removing it from the original location. The copied data can be placed in a different location by using the Paste menu option before copying something else.	Ctrl + C
Copy link	Copies the selected selection tree object as a link. Should be used for selection tree items located under Definitions. For more information regarding links, see the Definition in Section 6.8 on page 98.	Ctrl + Shift + C
Paste	<p>Inserts the previously copied part of the tree at the selected insertion point.</p> <p>There is a default behavior for how the copied element is pasted. For example, a condition is always placed as a child to a selected node.</p> <p>Enabled if a copy command has been done and elements are possible to insert at selected point.</p>	Ctrl + V
Paste...	<p>Inserts the previously copied part of the tree at the selected insertion point.</p> <p>If it is possible to paste the copied tree element both as child and sibling to the selected element, a menu will popup allowing the user to choose how the paste should act.</p> <p>Enabled if a copy command has been done and elements are possible to insert at selected point.</p>	Ctrl + Shift + V



Table 43 The Selection Tree Menu Options and Their Descriptions

Menu option	Description	Hot key
Delete	Deletes the selected part of the tree with all its content. The deleted data cannot be inserted elsewhere.	Delete
Import	Imports a selection tree configuration from file, see Section 9.1 on page 203.	Ctrl + I
Export	Exports a selection tree to file, see Section 9.2 on page 204.	Ctrl + E
Save	Saves the current selection tree. To associated ERE – saves the Selection Tree to the ERE the user is working on. To multiple EREs – gives the possibility to select several EREs to distribute the Selection Tree to.	Ctrl + S Ctrl + Shift + S
Save As	Opens a dialog that allows creation of a new Rating Period which holds a copy of the current Selection Tree.	-
Show References	Opens a window displaying all references to where the generic data is used in the selection tree. For more information regarding links, see Section 6.8 on page 98.	Ctrl + R
Find	Opens the Find application window. The find function is further described in Section 9.5 on page 211.	Ctrl + F
Goto Link Target	Locates the source of a selection tree object link. This menu option is only visible when targeting a linked object in the selection tree.	Ctrl + T
Move up	Moves the selected item one step up in the selection tree.	Ctrl + arrow up
Move down	Moves the selected item one step down in the selection tree.	Ctrl + arrow down



Table 43 The Selection Tree Menu Options and Their Descriptions

Menu option	Description	Hot key
Undo	<p>The Undo command reverses the last action in the Selection Tree.</p> <p>The undo action reverses:</p> <ul style="list-style-type: none"> • An added or removed item in the selection tree • Move operations in the tree (move up or move down) • Change a parameter of a condition or modifier • Import of a selection tree <p>Enabled if any action has occurred.</p>	Ctrl + Z
Redo	<p>The Redo command recreates the last undo action in the Selection Tree.</p> <p>The redo action recreates:</p> <ul style="list-style-type: none"> • An added or removed item in the selection tree • Move operations in the tree (move up/move down) • Change a parameter • Change a parameter of a condition or modifier <p>Enabled if any action has occurred.</p>	Ctrl + Y

6.3.2 Multiple Select

The selection tree supports selecting multiple nodes at the same time, see Figure 47.

The following actions are possible with multiple select:

- Copy nodes
- Copy links to nodes
- Delete nodes

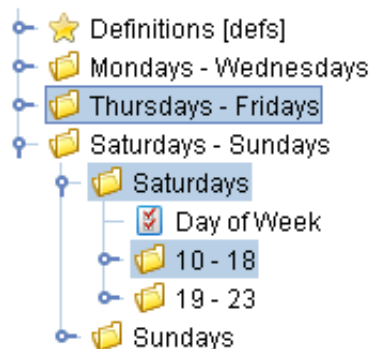


Figure 47 Example of Multiple Select

6.3.3 Drag-and-Drop

It is possible to move elements, copy elements, or create links to elements with drag-and-drop operations in the Selection Tree Editor. Confirmation of drag-and-drop move operations can be selected as an option, see Section 4.2 on page 19.

Use drag-and-drop with the left mouse button to perform the operation to the insertion point. The insertion point is shown by either a line in the structure or a highlighted object. Minimized nodes will expand when they are highlighted.

The following drag-and-drop operations are available within a rating period:

- Switching order between nodes, conditions, or modifiers, on the same level in the structure.
- **Ctrl** + drag-and-drop copies the selected element to the insertion point.
- **Ctrl** + **Shift** + drag-and-drop creates a link from the selected elements in the tree, to the insertion point.
- Moving a node into another node, making it a sub-node.

The following drag-and-drop operations are available between rating periods:

- Copying elements from one rating period to another.

Note: The visualized insertion point is not always the target in the structure. Certain restrictions and prioritizations apply to rating period structures. The insertion point will be at the closest target available.

6.3.4 Speed Typing

When a Node has been created in the Selection Tree, the property **Name** is automatically selected to enable direct edit of the node name.



The Speed typing functionality can also be triggered for any element in the Selection Tree by pressing the key **F2**. An exception is Links, that has no Name-property.

6.4 Common Selection Tree Input

The parameters in Table 44 are the setting for all selection tree elements, such as the root of the tree, conditions, nodes and modifiers. More settings may be available depending on the used element.

Table 44 Selection Tree Input

Field	Description	Range
Name	It is possible to choose a name for the root, definitions, and nodes. The names of conditions and modifiers are static and can therefore not be changed. If the name of the root is changed it affects the name of the Rating Period also. The change of the Rating Period name is introduced when saved.	Text field that can contain any keyboard characters.
Id	An identification of generic data to be used in the selection tree. The text is shown after the node condition or modifier within angle brackets. The text string must be unique within a selection tree.	Text field that can contain any keyboard characters.
Tag	The conditions and modifiers that may be updated using the Multi data Update tool, see Section 9.7 on page 222, have this parameter. Several elements may share the same tag, where updating the value of a data parameter affects all the elements with the same tag at the same time. It is recommended that the tag be a descriptive text, in order to keep track of the data easily.	Text field that can contain any keyboard characters. Up to 100 characters.
Comment	It is possible to add a comment for a node, condition, modifier, or a link. The comment is visible in the tooltip when the mouse pointer is placed over an entry in the selection tree. This gives the rating designer the possibility to explain the purpose or reasons an entry in the selection tree with more words than is offered when naming the node.	Text field that can contain any keyboard characters.

6.5 Selection Tree Validation

The parameters in the plug-ins used in the Selection Tree validate input.

6.5.1 Restrained Input

Input Fields in parameters that only take certain data type input, such as only integer numerical, restrict all non-numeric input, such as letters, see Figure 48. The application beeps and shows a message in the Status Bar of RMA and does not accept the entered value.

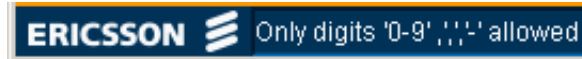


Figure 48 Only Numerical Data Type Input Allowed

6.5.2 Ranges

Some parameters only take numerical input of a certain range, such as short. If the entered value exceeds the range, the GUI beeps and shows a status message with the entered value and valid range, see Figure 49. If the parameter is missing, the parameter field and the tree turn red.

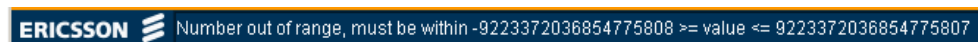


Figure 49 Only Numerical Data Type Input Allowed

6.5.3 Faulty Input

If the entered value has a proper format but has a faulty form, the parameter turns red as well as the tree element where the fault is, in the tree and the path from the root, see Figure 50.

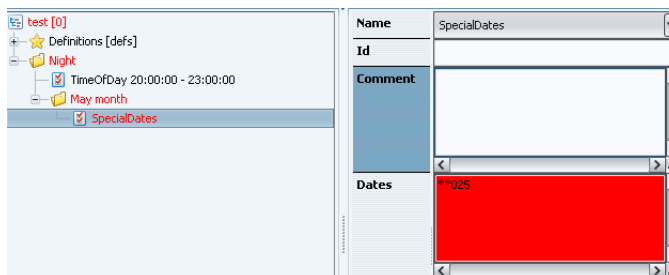


Figure 50 Faulty Input Turns the Parameter and the Tree Path Red

6.6 Selection Tree Root

The root of the tree is reached by clicking at the root element, marked with the name of the tree and [root]. Setting possibilities are reached through the Settings tab to the right in the window. The root of the tree contains parameters that affect the whole Selection Tree, such as the service used and the editor for TDFs, see Figure 51.

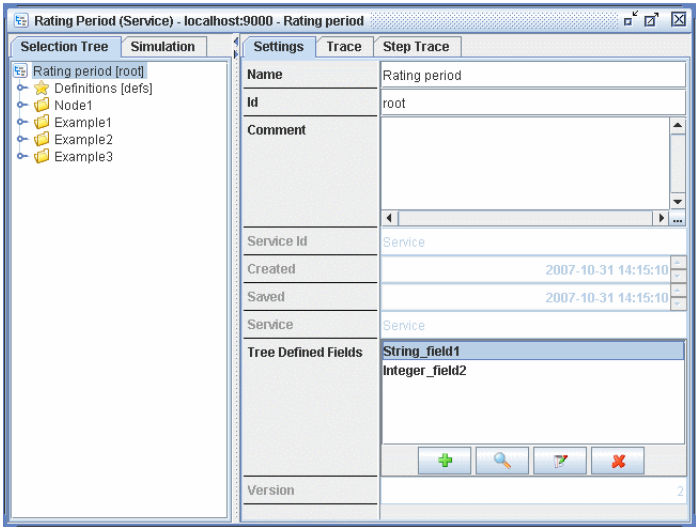


Figure 51 The Tree Root

6.6.1 Common Input Parameters

Table 45 shows common properties for input in the Selection Tree root. Some of the properties may not be included in some solutions where ERE has been integrated.

Table 45 Common Input in Selection Tree Root

Parameter	Description
Service Id	Shows the Service that is connected to this Selection Tree. Not possible to edit.
Created	Shows when the Rating Period was created.
Saved	Shows when the Rating Period was saved.
Service	Shows the Service that is connected to this Selection Tree. Not possible to edit.
Version	Shows the version of the tree

6.6.2 Tree Defined Fields Editor





The Tree Defined Fields editor is available in the root of all Selection Trees, see Figure 51.

The layout of the Tree Defined Fields editor is explained in Table 46.

Table 46 Tree Defined Fields Editor Layout

	Add a new Tree Defined Field
	View the selected Tree Defined Field

Table 46 Tree Defined Fields Editor Layout

	Edit the selected Tree Defined Fields
	Delete the selected Tree Defined Field
	Import file containing Tree Defined Fields
	Export Tree Defined Fields to file

Options available in the Tree Defined Field editor are also possible to find by right - click on a field in the list of created TDFs. This shows the **Add value**, **View value**, **Edit value**, and **Delete value** options. Export and import operations of TDFs have to be done through the buttons in the Tree Defined Fields editor shown in Table 46.

6.6.2.1 Creating a New Tree Defined Field

Click the **Add a new Tree Defined Field** button to add a new TDF in the editor, and an editing dialog for creating new TDFs appears, see Figure 52.

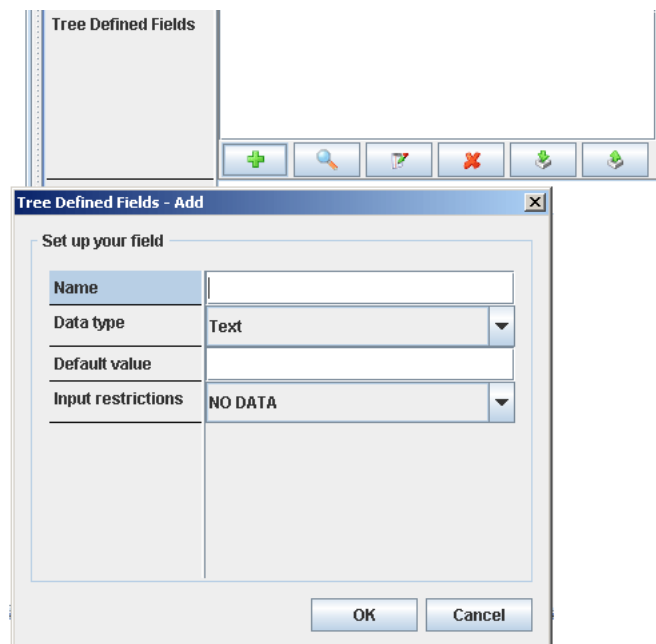


Figure 52 The Add and Edit Dialog

The following options are available for creating a new TDF:

— Name

It is mandatory to have a name on a TDF, so this text field cannot be empty. The name must be unique. The name cannot have more than 200 characters.



— Data Type

The data type to use for the TDF is selected in the drop-down dialog, see Figure 52. For available data types, see Table 47. If a data type is not chosen for the field, the TDF is created as a Text field.

Table 47 Allowed Data Types

Data type	GUI name	Allowed characters	Range	Default value
Amount	Amount	Numerical and a dot	0 - 999999999999999	0
Boolean	Boolean	True or false		False
Date	Date	0 - 9 and hyphens in the format YYYY-MM-DD		Current date
Floating Point	Floating-point number	Numerical and a dot		0.0
Long	Number	Numerical	Minimum value: -9,223,372,036,854,775,808 Maximum value 9,223,372,036,854,775,807	0
String	Text	All		
Time	Time	0 - 9, hyphens, and colons, in the format YYYY-MM-DD hh:mm:ss		Current time and date

— Default value

Default value is an optional parameter. The default value is used if the user wishes to have a default or start value for the TDF other than the ordinary for the given data type, see Table 47. If the entered value is out of the minimum or maximum bounds for the selected data type, a warning is given in the status bar, it is not possible to enter the faulty value.

If a range is selected for a TDF of the type Number, the Default field will not have this restriction. This to enable the possibility to have a start value outside the range which represents that no value has been set.

The TDF is created with the submitted default value. The submitted value is not considered as default value in, for example, the **CompareField** condition or Set Field modifier, when the TDF is used there.

- Input restrictions

This is an optional choice. Input restrictions are used if the user wishes to restrict the value that can be set for the TDF in the **CompareField** condition, modifier Set Field, and in the Simulation Panel. It is possible to set a range and also define exactly what values can be chosen.

NO DATA – Default value, input is not restricted in any other way than what values the chosen data type can take, see ranges in Table 47..

ENUMERATION – With enumeration of the exact values for this TDF. A field with a button Edit appears right under Simulation Data when Enumeration is selected. By clicking **Edit** a dialog frame appears with edit possibilities for what values should be possible for this TDF, see Section 5.3.6.2.2 on page 44.

In the editor it is possible to add more rows, delete rows and reset all.

RANGE – With the range option it is possible to choose a range within what bounds values is possible to be set for the TDF. A Min and Max field appears right under the Simulation Data in the TDFs Edit dialog when Range is selected. By filling in the Min and Max field the exact range of allowed values for this field are specified. It is also possible to use the range of the data type long, by selecting the Min and Max box. The Min and Max fields are automatically filled with the minimum and maximum value of data type long. This is the same as not selecting a range for the field at all. The range option is available for the data type Number.

If the number submitted Max or Min value is out of range, an error message is shown in the lower left part of the RMA GUI, see Figure 53.

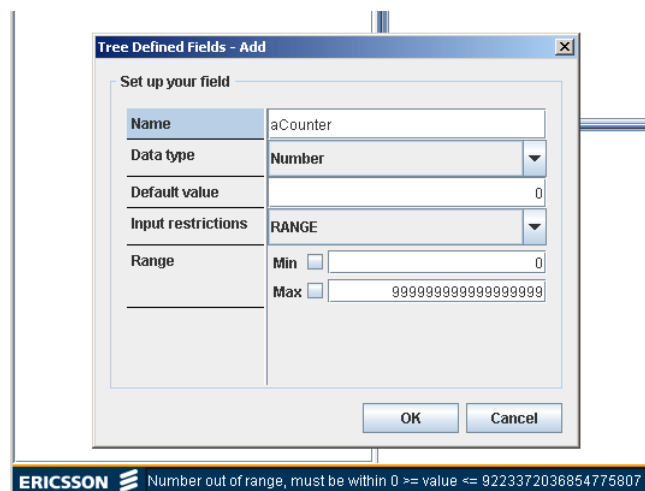


Figure 53 Max Value is Out of Range and an Error Message is Displayed

6.6.2.2 Editing Existing Tree Defined Fields

After creation of a TDF, it is possible to edit the configuration of a chosen TDF.



Click the **Edit the selected Tree Defined Fields** button to edit a TDF. A dialog with all configurations appears. Mark the field that should be changed in the Tree Defined Fields editor. It is possible to change all values, as described above in how a new TDF is created, in section Section 6.6.2.1 on page 80.

If the user selects a field in the list that is used in a condition or modifier in the tree, the possibility of editing is constrained to adding rows to the Enumeration. The rest of the fields are locked from editing.

In the Enumeration editor it is possible to add additional rows, change the name of the existing rows in the enumeration but existing values are locked from editing. It is not possible to delete rows that are already loaded, only rows that were added after editing can be deleted.

6.6.2.3

Viewing a Tree Defined Field

Click the **View the selected Tree Defined Field** button to view selected TDF. It is not possible to edit the TDF in the dialog window. This option is also available for fields that are in use in the Selection Tree.

A double-click on a TDF, in the list also opens a view dialog window, see Figure 54. The purpose of the view option is to enable the possibility of looking at the setup of a TDF that is locked for editing.

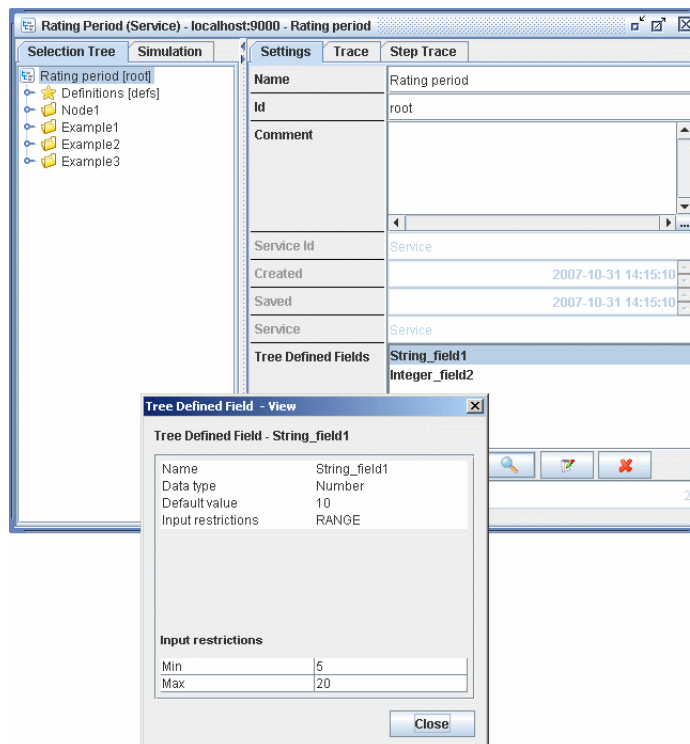


Figure 54 Tree Defined Field View Dialog

6.6.2.4 Deleting a Tree Defined Field

TDFs can be deleted through the Tree Defined Fields editor with the **Delete the selected Tree Defined Field** button. It is also possible to delete a TDF by using the Delete button on the keyboard. The user will be asked to confirm the removal of the TDF. If the field is in use in the Selection Tree, the delete button is disabled and therefore this operation may not be performed.

6.6.2.5 Import of Tree Defined Fields

Click the **Import file containing Tree Defined Fields** button to import TDFs into the root of the current Selection Tree. Select the folder containing the TDF file by clicking the **Browse** button, and select the file. Click the **Next** button to select TDFs to import, see Figure 55. Click the **Next** button again to start the import. The result is shown ERE Wizard.

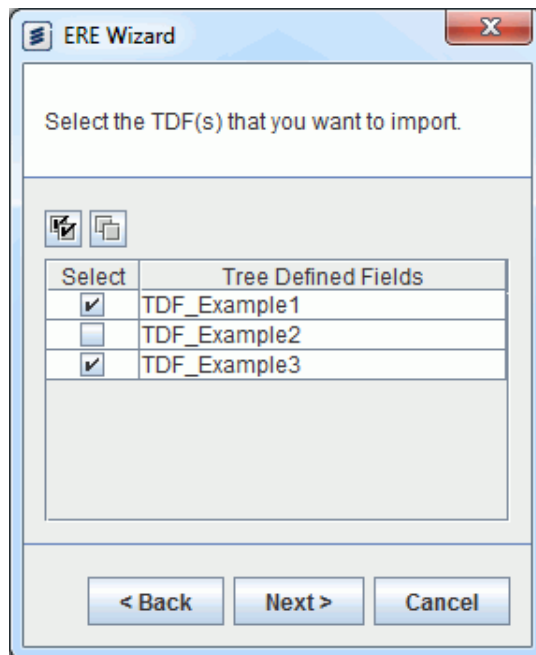


Figure 55 Selected Tree Defined Fields

6.6.2.6 Export of Tree Defined Fields

Click the **Export Tree Defined Fields to file** button to export TDFs from the root of the current Selection Tree. Select a destination folder by clicking the **Browse** button, and enter a filename. Click the **Next** button to select TDFs to export. Click the **Next** button again to start the export. The result is shown in the ERE Wizard, see Figure 56.

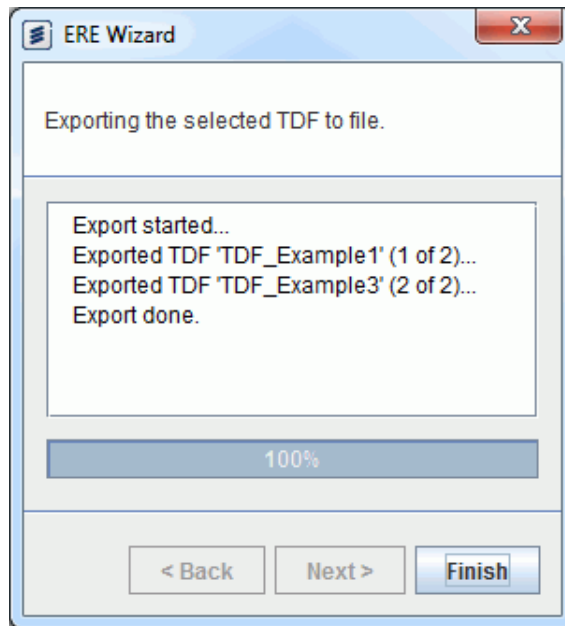


Figure 56 Tree Defined Fields Export Result

6.6.2.7 Comparison of Tree Defined Fields in Selection Tree Difference Detection Tool

The **Selection Tree Difference Detection Tool** considers only mandatory parameters of TDFs during comparison.

6.7 Special Input Editors

Other editors in the GUI that are not explained elsewhere in this User Guide are described in the following subchapters.

6.7.1 Array Editor

An array collection can contain indexed series of values. There are two editors for array collections in simulation.

6.7.1.1 Array Editor for Data Fields

An array field can contain indexed series of values, these are called array collections. A value in an array collection is represented and referred to by its index number. The array editor used for data fields is illustrated in Figure 57, and further explained in Table 48.

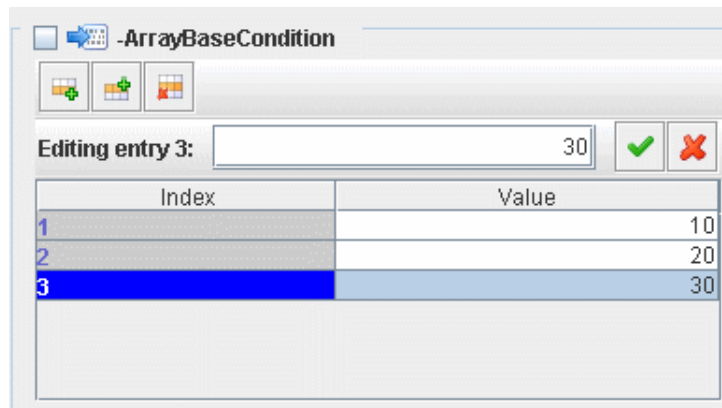


Figure 57 Data Field Array Editor

Configuration fields that are represented as an array are edited using a table editor. Generally, the first column is locked for editing since its purpose is to show the row number. The other column may be edited with the values of the data type associated with the field.

Table 48 Explanation of the Data Field Array Editor

Button	Description
	Adds an empty row below the selected row.
	Adds an empty row above the selected row.
	Deletes the selected row.
	Open an additional editor for data types that require one. This button is displayed only when additional input is allowed.
	Commits the value to the array.
	Cancels the editing and leaves the array unchanged.

If the data type associated with a field requires an editor to be opened, for example, the field may be a bit pattern, choose the **Click to edit** button. This opens an editor where the values can be selected. The buttons to commit the value to the array and to cancel the editing are available in the new editor. The values from the editor are first available in the array when they are committed using the appropriate button.

6.7.1.2 Array Editor for Hierarchical Fields

In this array editor, values are entered into the index of one or more associated data fields. The values entered are ascendant indexed, but the index is not visible in the editor. The layout is further explained in Table 49. In Figure 58, the value 234123 is entered into the data field **MSISDN**, and the value 19511223 is entered into the data field **DoB**. Both values are indexed as 0.

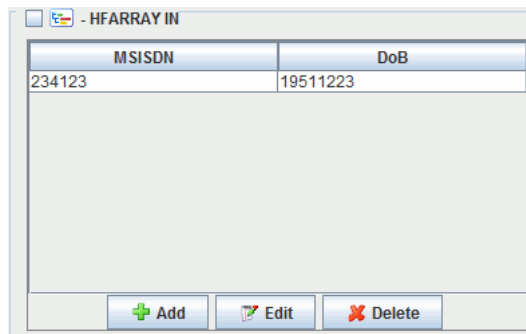





Figure 58 Hierarchical Field Array Editor

Table 49 Explanation of the Hierarchical Field Array Editor

Button	Description
 Add	Opens up a dialog window to input data into the associated fields.
 Edit	Edit the selected row.
 Delete	Deletes the selected row.

6.7.2 Map Editor

Similar to the array editor, the map editor allows data to be edited into a map collection. There are two editors for map collections in simulation.

6.7.2.1 Map Editor for Data Fields

A data field map collection contains a field with values, each represented by a unique key value. A value in a map collection is represented and referred to by its key.

The editor is similar to the array editor for data fields, but each value is represented, and referred to, by its unique key. The key can be set to any value in string format, in the **Name** column, see Figure 59.

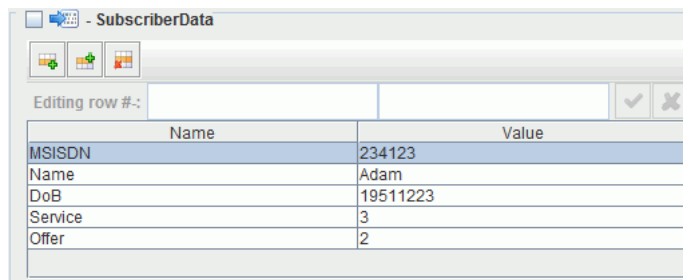


Figure 59 Data Field Map Collection

6.7.2.2 Map Editor for Hierarchical Fields

In this map editor, values can be entered into one or more associated data fields. The key is represented by at least one predefined key field. If more than one field is a key, these fields are combined to create a composite key for the map collection. Key fields are indicated by a key symbol, see Figure 60.

MSISDN	Name	DoB	Service	Offer
234123	Adam	19511223	3	2

Figure 60 Hierarchical Field Map Collection

6.7.3 Set Editor

The set editor allows input for a set collection. There are two editors for set collections in simulation.

6.7.3.1 Set Editor for Data Fields

The data field may be edited to contain non-indexed unique values. The data field set collection in Figure 61 contains three unique values.

Value
Green
Yellow
Red

Figure 61 Data Field Set Editor

6.7.3.2 Set Editor for Hierarchical Fields

In this set editor, values are entered in the associated data fields. If only one data field is associated, the value has to be unique. If multiple data fields are associated to the set collection, the combined value has to be unique.



MSISDN	DoB
234123	19511223

Figure 62 Hierarchical Field Set Editor

6.7.4 Editor for Data Field Collection in Hierarchical Field Collection

Hierarchical field collections can contain data field collections. In this case, a combination of the data field editor and collection editor is displayed in the same dialog window, when adding a new entry. The rules for each entry, when a data field collection is used in a hierarchical field collection, applies as for the respective editor.

For example, when a data field set collection is used in a hierarchical field array collection, each entry consists of a combination of the hierarchical field collection and any of the entries in the data field collection, see Figure 63.

Name	ID	Descriptions
Donald Duck	42	<SET>
Daisy Duck	44	<SET>
Huey	51	<SET>
Dewey	52	<SET>
Louie	53	<SET>

Figure 63 Data Field Set Collection in Hierarchical Field Array Collection

Pressing the **Add** button brings up an editor consisting of a combination of a data field set collection and a hierarchical field array collection.

The values in the data field collection are never displayed in the hierarchical collection, they can be seen by pressing **Edit** when an entry is selected, see Figure 64.

Edit details		
Name	Donald Duck	
ID	42	
Descriptions	<div> <div> </div> <div> Editing entry <input type="text"/> <input type="checkbox"/> <input type="checkbox"/> </div> <div> Value <div> short temper positive annoying </div> </div> </div>	

Figure 64 Edit of Data Field Set Collection in Hierarchical Field Array Collection

6.7.5 Array Group Editor

The array group editor, see Figure 65 enables the user to edit related data fields like arrays in different columns in a table.

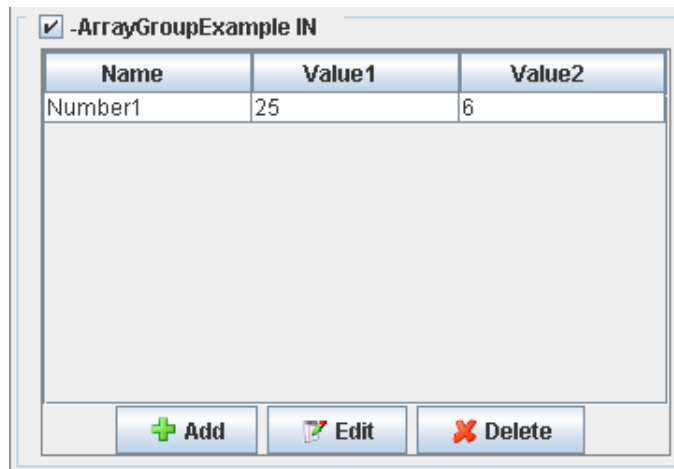


Figure 65 Array Group Example

The editor has three buttons at the bottom allowing creating, editing, and deleting rows in the data table, see Table 50.

Table 50 Buttons in the Array Group Editor

Icon	Action	Description
	Add	Adds a new row
	Edit	Edits the selected row in the table
	Delete	Deletes the selected row in the table

When a row is added or edited a popup window appears and a value for every data field in the group can be inserted, see Figure 66.

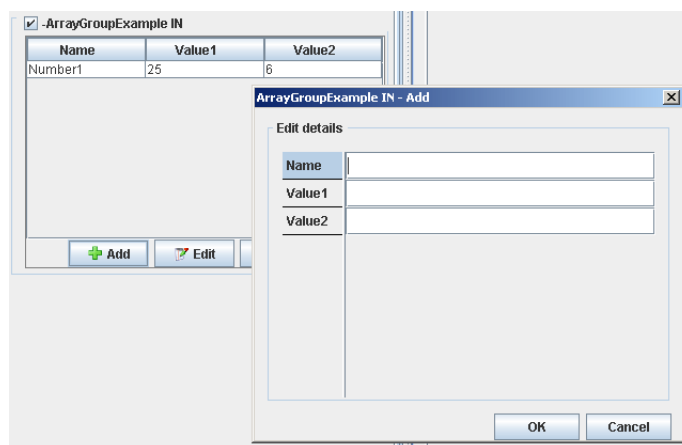


Figure 66 Adding a New Row to Array Group

When the **OK** button is pressed, the values are set in the array group editor. The rows of the associated fields are indexed in the ascending order, starting from zero. In Figure 67, the rows are indexed 0-1.



Name	Value1	Value2
Number1	25	6
Number 2	200	10

Figure 67 Array Group Values

The array group is configured in the Service Definition, see Section 5.3.5 on page 39.

6.7.6

Map Group Editor

The map group editor looks the same as the array group editor. The only, non visible, difference is that the rows in the associated fields have key values.

The map group editor looks the same as the array group editor. The only visible difference is that the rows in the associated fields have key values, indicated with a key symbol. See Figure 68.

Name	Rating Plan	Bonus Group
Free SMS	1	3
Free Voice	2	1

Figure 68 Map Group Values

The map group is configured in the Service Definition, see Section 5.3.5 on page 39.

6.7.7

Bit Pattern Editor

The Bit pattern editor, see Figure 69, enables the user to edit a numeric field by setting or not setting its bits. Can be used for any numeric field by setting the



Simulation data option to BIT PATTERN in the Service Definition, see Section 5.3.6.2.3 on page 44. The editor is a panel containing the bits, which are possible to edit, together with a text field for the unsigned decimal representation of the field value.

In the simulation panel, the editor is folded and is visible when expanded.

The components of the Bit pattern editor and the descriptions of them are presented in Table 51.

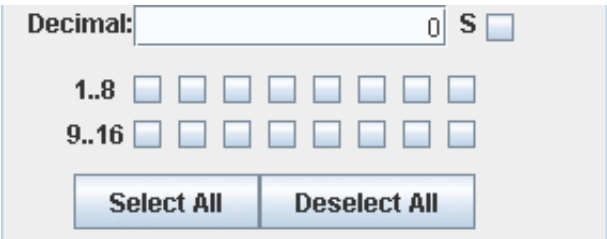


Figure 69 Bit Pattern Editor

Table 51 Components in the Bit Pattern Editor

Components	Description
Text Field	Shows the current bit setup in unsigned decimal form. The field can also be edited, causing a direct update of the bit setup.
S-check box	This check box gives an option of how the bit panel should be arranged. The alternatives are Normal and Scientific mode. In Normal mode, when the check box is cleared, the bits are shown starting with the first bit and ending with the last. This is the default choice. In Scientific mode, when the check box is checked, the bits are shown in reversed order, for example, starting with the last bit and ending with the first.
Bit setup panel	Shows all bits that are possible to edit. When a bit is edited, the new value is calculated and shown in the text field in unsigned decimal form.
Select all	When pressed, all bits are set.
Deselect all	When pressed, all bits are cleared.

6.7.8 Octet String Editor

The octet string editor, see Figure 70, allows input of a string that is restricted to containing either ASCII or hexadecimal values. The drop-down menu indicates if the input should be Hex or ASCII. If a change is made in the drop-down menu, the input value changes as well. When defined as Hex, only hexadecimal digits are allowed (0-9, A-F).



Figure 70 The Octet String Editor

6.7.9 Date Editor

The Date editor, see Figure 71 , enables to enter a specific date. The date can either be entered manually or by opening the Date picker panel. If entered manually, the date format should be YYYY-MM-DD. The years ranges from 0000 to 9999, the month from 01 to 12 and the days from 01 to 31.



Figure 71 Date Editor

The Date picker panel, see Figure 72, enables picking a specific date from a calendar, where the year can be selected through a spinner, the month from a drop-down and the day from picking the day in the calendar. The Date picker panel is reached by pressing the calendar button in the Date editor. The year in the year spinner can be increased or decreased by pressing the arrow up or down, alternatively by entering a year. Today's date can be selected by pressing **Today**. The date is submitted by pressing **OK**, if pressing **Cancel** the changes will be discarded.

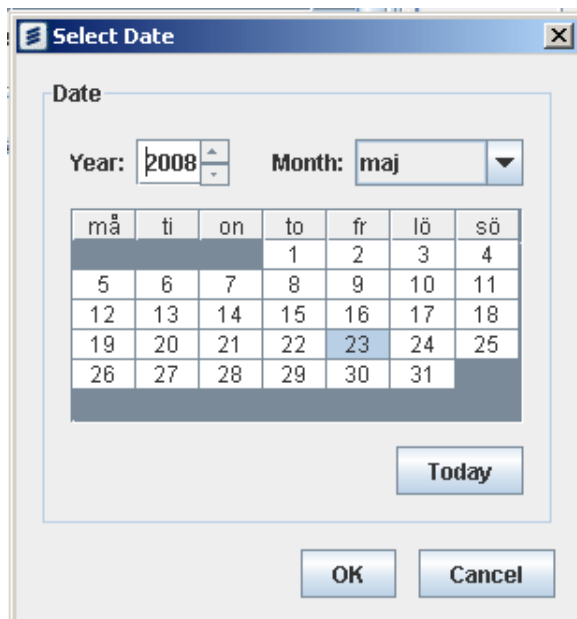


Figure 72 Date Picker Panel

6.7.10 Relative Dates Chooser

This allows an expression to be set up, a defining a date to be offset, relative to another specified date.

The editor is shown when defining a field of the type date, for example for use with the **CompareDateWithMask** condition. The editor can be seen in Figure 73.

The parameters the editor are described in Table 52.



Table 52 The Parameters in the Relative Date Chooser

Name	Description	Values
Use Field	If the check box is selected, the Date field contains a drop-down menu of the applicable date fields.	
Date	The date that the resulting date is relative to. It may be the value of a field, or entered manually by typing or by selecting from the date picker panel.	The date format is YYYY-MM-DD.
Operation	Defines if the relative date should be before or after the date in the Date field.	One of: <ul style="list-style-type: none"> NONE ADD SUBTRACT
Offset	Defines the magnitude of the difference between the two dates.	An integer value, plus one of: <ul style="list-style-type: none"> Days Months Years

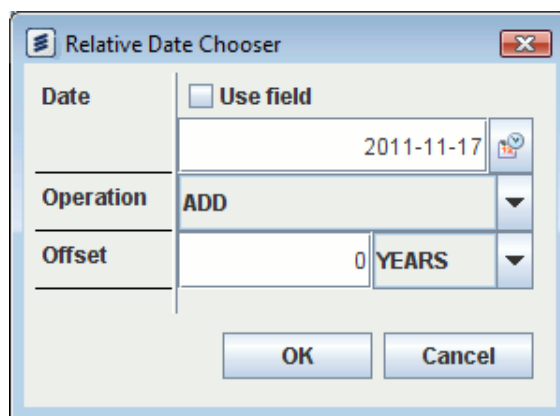


Figure 73 Relative Date Chooser Panel

6.7.11 Relative Time Chooser

This allows an expression to be set up, defining a time to be offset, relative to another specified time.

The editor is shown when defining a field of the type date, for example for use with the **CompareField** condition. The editor can be seen in Figure 74.

The parameters the editor are described in Table 53.



Table 53 The Parameters in the Relative Time Chooser

Name	Description	Values
Use Field	If the checkbox is selected the Date field contains a drop-down menu of the applicable date fields.	
Date	The time that the resulting time is relative to. It may be the value of a field, or entered manually by typing or by selecting from the date and time editor panel.	The format is YYYY-MM-DD hh:mm:ss
Operation	Defines if the relative time should be before or after the date in the Time field.	One of: <ul style="list-style-type: none">• NONE• ADD• SUBTRACT
Offset	Defines the magnitude of the difference between the two times.	An integer value, plus one of: <ul style="list-style-type: none">• Seconds• Minutes• Hours• Days• Months• Years

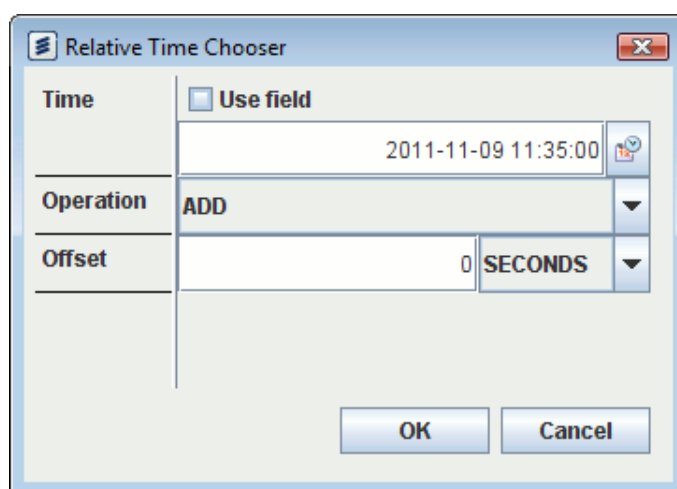


Figure 74 Relative Time Chooser Panel



6.7.12 Time Editor

The Time editor, see Figure 75, enables to enter a specific time in the format hh:mm:ss with constraints on hours from 00 to 23, minutes from 00 to 59, and seconds from 00 to 59. The Date can also be entered through the Time picker panel, and is reached by pressing the calendar button in the Time editor.



Figure 75 Time Editor

The Time Picker Panel, see Figure 76, enables entering a time in the hour, minute and second fields or selecting the present time by pressing **Now**.

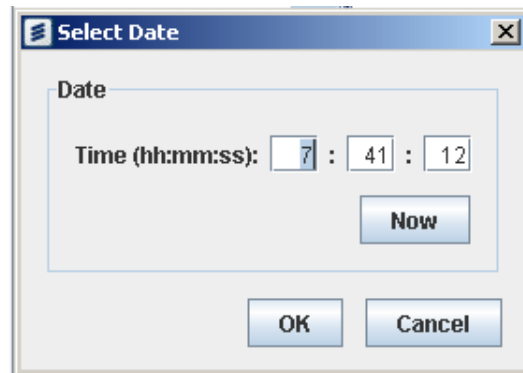


Figure 76 Time Picker Panel

6.7.13 Date and Time Editor

The Date and Time editor, see in Figure 77, enables selecting a specific date and time. The date and time can either be entered manually or by opening the Date and Time picker panel. If entered manually, the format should be YYYY-MM-DD hh:mm:ss. The years ranges from 0000 to 9999, the month from 01 to 12, the days from 01 to 31, the hours from 00 to 23, minutes from 00 to 59, and seconds from 00 to 59.



Figure 77 Date and Time Editor

The Date and Time picker panel, see Figure 78 is reached by clicking the calendar button by the Date And Time Editor. The panel allows editing the date by selecting a year from the spinner, a month from the drop-down list, and a date from the dates shown. The panel also allows editing the time by specifying an hour, a minute, and a second from the corresponding fields. By selecting **Now**, the current date and time is selected.

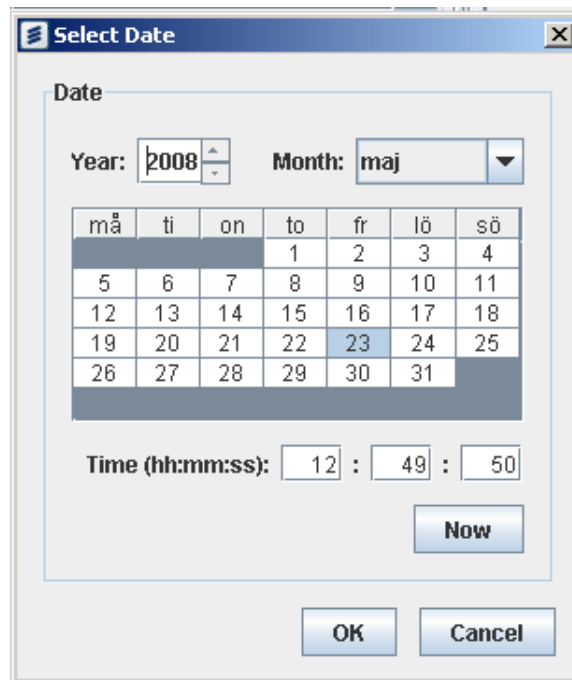


Figure 78 Date and Time Picker Panel

6.7.14 Field Chooser

In the selection tree editor, fields can be selected as targets for plug-ins. The target field can be selected from a filterable drop-down list. When typing in the text box, only fields containing the sustained characters are displayed in the drop-down list. Fields can be given a tooltip at definition which is displayed when placing the mouse pointer over the field's name in the list. Figure 79 shows how a field chooser drop-down list that has been filtered by **Date** and the field **DateEnum** has been defined with a **Tooltip**, Restricted by policy.

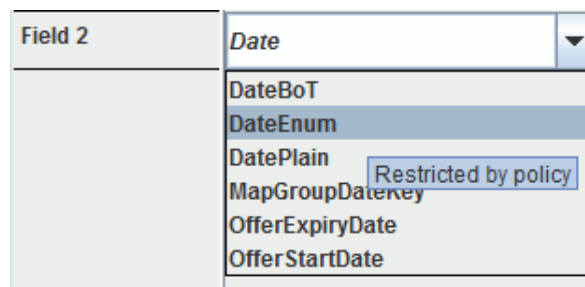


Figure 79 Field Selection

6.8 Definitions Element in the Selection Tree

In the Selection Tree it is possible to define generic nodes, conditions, and modifiers that can be used in many places in a Selection Tree. These tree elements



can be referred to from places in the tree by linking to them, see Figure 80. Links between this type of generic data should be used when the data is to be used in more than one place in the Selection Tree. The advantage is that the generic data only needs to be updated in one place for the entire Selection Tree and redundant information is avoided.

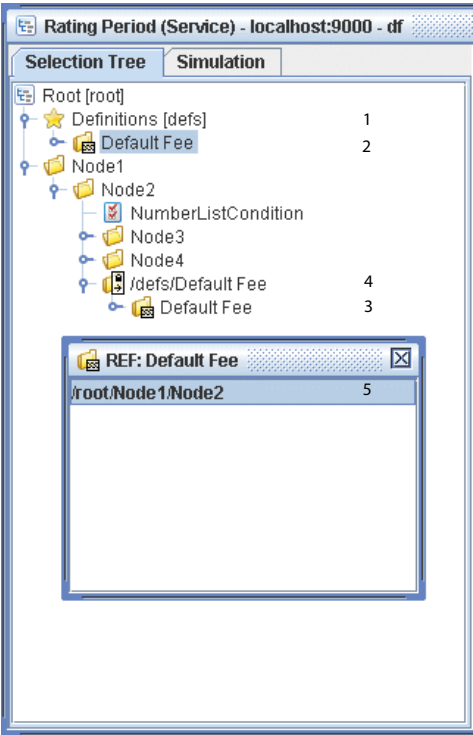


Figure 80 Reference to Item

Nodes, conditions, and modifiers that should be generic are added under Definitions. Tree elements that are linked to are marked by a link icon. Descriptions of the icons, used for linking, are found in Table 54.

Table 54 Description of Icons Used for Linking

	Description
1	Definitions All selection tree entries placed under this node icon are generic definitions. Nodes, conditions, and modifiers can be used as definitions.
2	Linked selection tree entry. The rating rule data in the node, condition or modifier with this icon is a generic definition. A definition with a link icon, means that something links to the definition. Tree elements under Definitions, that are not marked with a link icon, is not used anywhere in the selection tree.



Table 54 Description of Icons Used for Linking

	Description
3	<p>Linked selection tree entry.</p> <p>The rating rule data in the node, condition or modifier with this icon is a generic definition.</p> <p>A definition with a link icon means that something links to the definition. Tree elements under Definitions that are not marked with a link icon are not used anywhere in the selection tree.</p>
4	<p>Selection tree data locked.</p> <p>In Figure 80 a locked node link icon is used but modifier- and condition links can also be locked.</p> <p>For the anchored selection tree entry, an arrow is displayed in the lower left corner, which indicates that the rating rule data is defined as generic data under Definitions. In this example, it uses the data defined in the section indicated as 3. It also displays a lock in the upper right corner, indicating that the data is locked for editing.</p> <p>Unlocking of data is performed by choosing the link icon and clearing the Locked check box.</p>
5	<p>References</p> <p>By selecting a used anchored selection tree entry and choosing Show References from the context-sensitive menu, an application window is displayed, showing all references to where the generic data is used in the selection tree. It is possible to jump directly to a referenced entry by clicking on it.</p> <p>If data is changed here it affects all instances where the data is used in the selection tree.</p>

It is possible to see existing links to an object in the selection tree. This is done by selecting **Show References** from the context-sensitive menu on the selected object, or by pressing **CTRL+R**. If the object has any links to it, a dialog window appears showing all the tree paths where it has been linked. If the object has no links, the option **Show References** is disabled in the context-sensitive menu, and no results will show when pressing **CTRL+R**. See Figure 81 for how **Show References** lists links to an object.

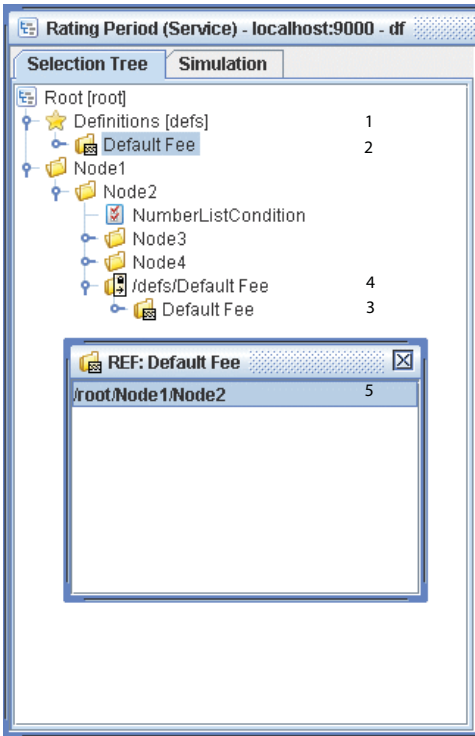


Figure 81 Show Reference

To find the source of a link, right-click the linked object to bring up the context-sensitive menu, select **Goto Link Target**. This locates the source object in the selection tree structure.

6.8.1 Common Parameters for Items in Definitions

The parameters described here relate to the creation of links in a Selection Tree, see Figure 82 and Table 55.

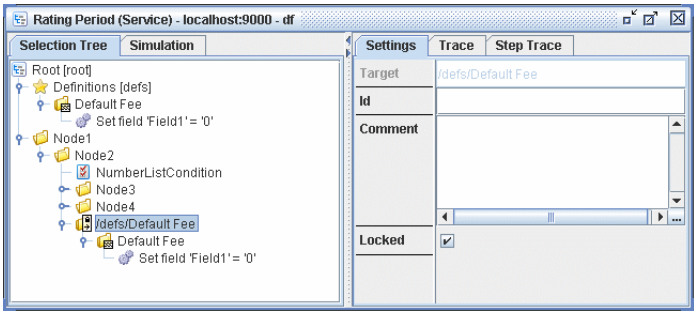


Figure 82 Locked Definition

The link Definition holds the data described in Table 55:



Table 55 Data Held in the Link Definition

Data	Description	Format
Target	A read only field where the target reference is shown. The target reference indicates where the link origin is located in the selection tree. Example: /defs/Default Fee.	
Locked	This is a check box that has to be cleared to be able to modify the generic rating data in the inserted link.	Check box checked – generic data locked for editing. Check box cleared – generic data can be edited.

For a link to work, the field **Id** has to be filled in for the generic node condition or modifier.

Note: A change in the generic data affects all instances where the data is used in the Selection Tree.

6.8.2 Using Definitions

An example of how a generic condition is defined and used in an existing selection tree is shown in Figure 83 and described in the following list:

1. Right-click on the **Definitions** icon.
2. Choose **Add child** condition from the context-sensitive menu.

A new condition is displayed followed by the name of a condition.
3. Select the new condition.

The condition configuration options are shown to the right of the application window.
4. Enter the generic condition values for the chosen condition type.
5. Enter a text string in the **Id** field. The activation of the data is done automatically. The **Id** text string is displayed after the condition within square brackets.
6. Right - click on the anchored condition.
7. Choose **Copy Link** from the context-sensitive menu.
8. Place the cursor on the insertion point in the selection tree where the generic condition should be used.
9. Right-click and select **Paste** from the context-sensitive menu.



A locked condition link icon is displayed followed by the link reference.

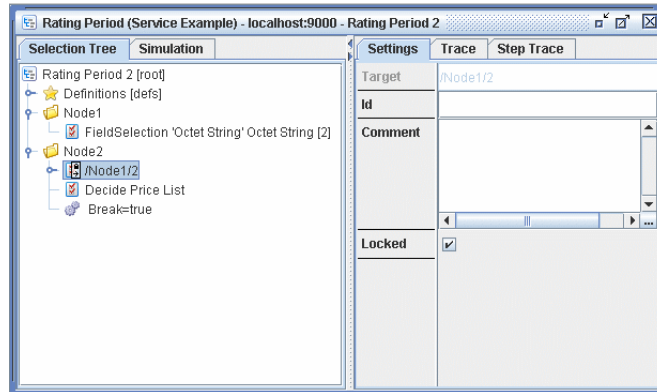


Figure 83 Example of Using Definitions

6.9 Selection Tree Analysis

The selection tree analysis always starts at the root, and the selection tree is then analyzed according to the following principles:

- When more than one node or condition exist on the same sublevel in the selection tree, it is analyzed in accordance with a logical **OR**. That means that one of the nodes or conditions must be true for the analysis to continue on that sublevel.
- To reach the next sublevel in the tree, the rating analysis compares all the overlaying nodes and conditions in accordance with a logical **AND**. That means that all the overlaying nodes and conditions must be true to reach the next sublevel.

The examples below show two selection tree analyzes performed under different circumstances, see Figure 84, Figure 85 and Table 56.

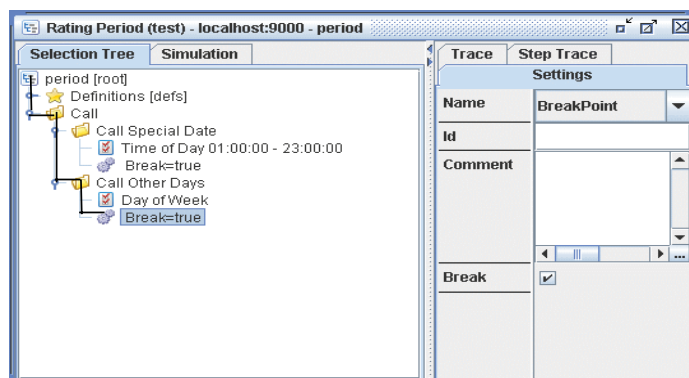


Figure 84 Example 1 of Selection Tree Analysis

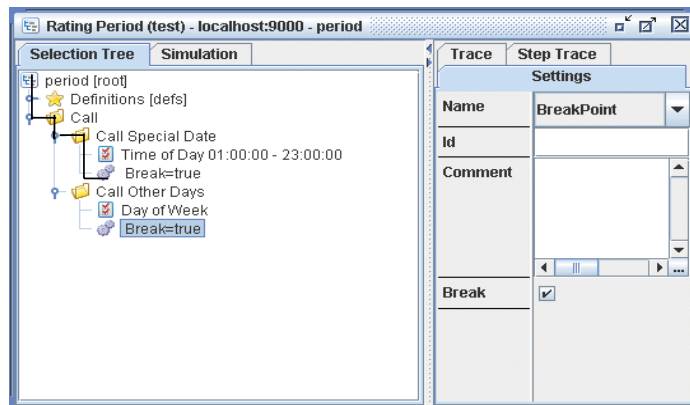


Figure 85 Example 2 of Selection Tree Analysis

Table 56 Example of Selection Tree Analysis

Example	
1	2
An MMS is to be rated	An SMS is to be rated
<p>In this example, the analysis goes into the Message node and finds that the service type is not SMS. It then performs a logical OR with the SMS node, and MMS node and comes to the conclusion that MMS is true.⁽¹⁾</p> <p>The analysis is then continued in the MMS node. A logical AND is performed on the conditions Decide Service Type, and Decide Price List and comes to the conclusion that it is true and the one hit fee is found and the analysis is finished.</p>	<p>In this example, the analysis goes into the Message node and finds that the service type is SMS. Therefore, no logical OR is performed and the analysis is then continued in the SMS node. A logical AND is performed on the conditions Decide Service Type and Decide Price List and comes to the conclusion that it is true and the one hit fee is found and the analysis is finished.</p>

(1) A logical **OR** is also performed if more than one condition are grouped on the same sublevel.

The information under the Common definitions icon, is used in the rating analysis when a Link to definition icon is found in the selection tree. For more information regarding linking, see Section 6.8 on page 98.

The conclusion of the selection tree analysis depends on which modifiers that are used and the characteristics of the solution or product where ERE is integrated.

6.10 Using Nodes, Conditions, and Modifiers

The logic of the selection tree is decided by how the different tree elements are used - nodes, conditions, and modifiers. This section describes how the logic of the tree is evaluated, and gives examples of how to build up a tree.



6.10.1 Using Nodes in the Selection Tree

The node is a logical container that holds the other elements as children.

Nodes are executed in order, from top to bottom. The conditions in a node form an expression. For the node to be true and the execution to continue to sub-nodes, at least one of the conditions in the node must be evaluated as true, see Figure 86.

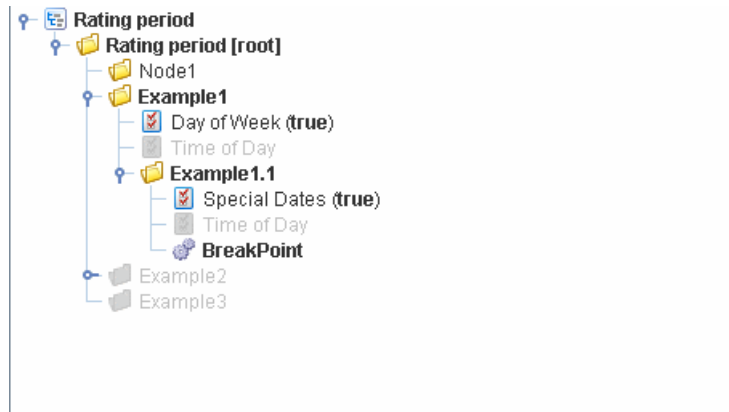


Figure 86 Condition True

If none of the conditions in the first node are evaluated as true, the execution continues to the next sibling node, see Figure 87. The execution of the nodes continues until a modifier that stops the execution, such as an Exit modifier, is hit or when the tree reaches the end.

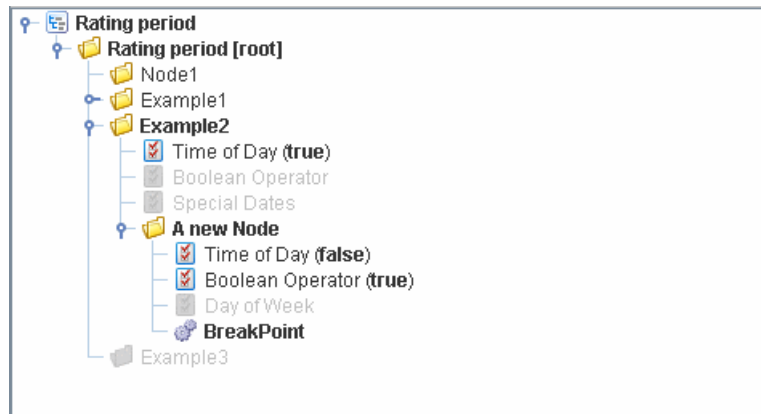


Figure 87 Condition False

Certain parameters are available for a selection tree node, and are described in Table 57:



Table 57 Parameters in a Selection Tree Node

Field	Description	Format
Type	Indicates if the node is a normal node, or an If/Else node, see Section 8.1.2 on page 144 for a description of the If/Else node.	Node or If/Else Default: Node
Inverted	Check box indicating if the conditions grouped by the node should be inverted.	Default: Unchecked

6.10.1.1 Adding a Node in the Tree

Nodes may be added to the selection tree on the current level in two ways.

They may be added as a sibling to another selection tree element. Select an element as insertion point on the level in which the new node is to be inserted, and choose **Add child > Node**. The new node will be inserted as a sibling to the element selected as insertion point, either immediately after it if the element is a node or modifier or immediately after all conditions if the element is a condition.

They may also be added as a sub-node to another node. To add a sub-node, select the node that is to be the parent node, and choose **Add child > Node**. The new node will be inserted as a child to the selected parent node, after all conditions but before any modifiers.

6.10.2 Using Conditions in the Selection Tree

Conditions are evaluation points in the tree. Conditions must be placed before modifiers or sub-nodes in a node. One condition can be placed alone under a node, but it is also possible to add several conditions before one or more modifiers or sub-nodes, see Figure 88.

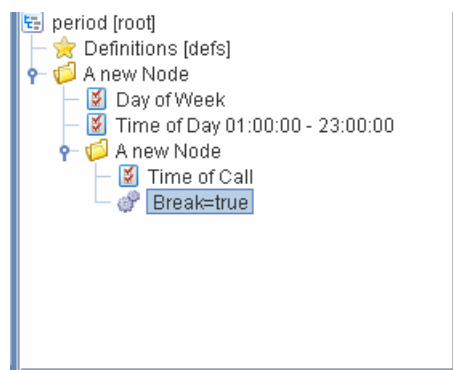


Figure 88 Conditions

If several conditions are used after each other in the tree, they may be considered as OR statements. When one condition in the node is evaluated as true, the



following conditions in the same level are not executed. Instead, the next modifier or sub-node is found and evaluated.

To construct an AND statement using conditions, place subsequent conditions in sub-nodes after the first condition, see Figure 89.

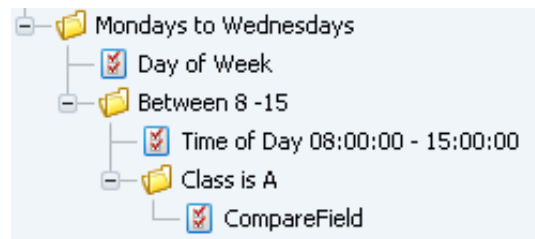


Figure 89 Constructing an AND Statement Using Conditions

6.10.3 Using Modifiers in the Selection Tree

Each logical path in the selection tree should end with at least one modifier. It is also possible to have several modifiers in the same level to be executed after each other, see Figure 90.

If a modifier that ends the tree execution, for example, Exit or Rate, is placed above one or more modifiers, the following ones are not executed. Therefore, it is important to consider the order of the modifiers. If the user wants to put several modifiers on the same level, as in the example below, make sure that only one is of the type that ends the tree executed and that it is placed last. Section 8.3 on page 183 describe which modifiers that end the tree execution.

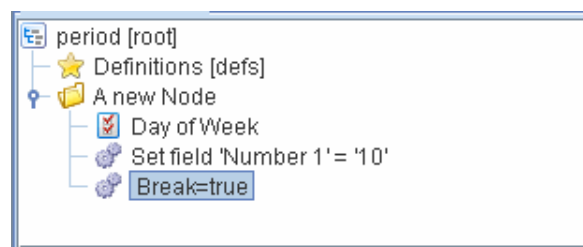


Figure 90 Two Modifiers

Modifiers can be inserted in a node or after conditions, but not as standalone. A modifier can be added on the same level as the insertion point or be added as a child, and be put in the substructure.

6.11 Selection Tree Qualifiers

Selection Tree Qualifiers (STQs) are a method of defining a path to a certain entry in a selection tree. Each node or subnode in a tree may be assigned a qualifier, which is a positive integer. The STQ of the node is set in the **Qualifier** field. As the tree is executed, these numbers are combined to give a unique path to the entry currently being evaluated. As an example, a node may be assigned the STQ 1,

the first subnode of this be assigned the STQ 14 and a further subnode may be assigned the STQ 45 and a modifier in this subnode be evaluated. The path to this modifier is thus 1,14,45 as the modifier is being evaluated. Figure 92 shows an example of a node, where the STQ has been set to 2.

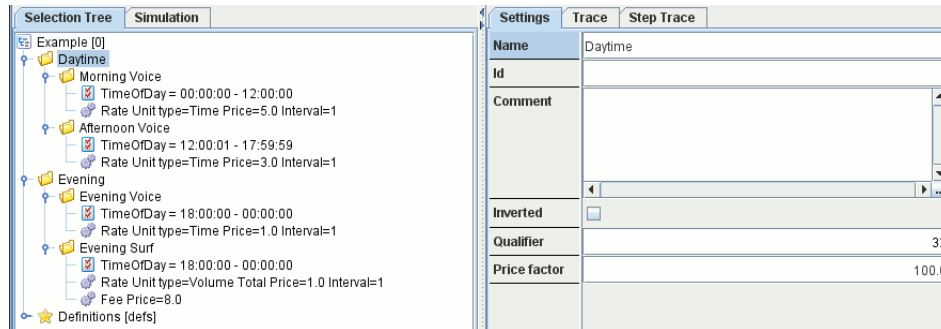


Figure 91 An Example of an STQ

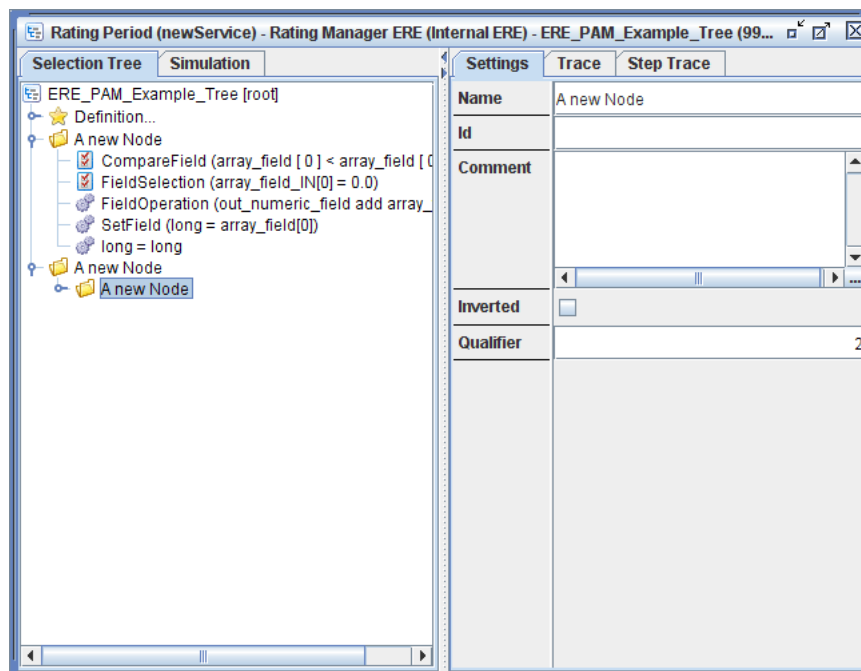


Figure 92 An Example of a Node with an STQ

Multiple nodes may have the same STQ, in which case the **Recycle** icon will be activated, indicating that the STQ number is being reused. Figure 93 shows this icon.



Figure 93 Recycle Icon

The STQ path to the modifier, condition or subnode being evaluated may be found in the trace while simulating the tree in question. Figure 94 shows the simulation trace for the STQ path laid out earlier in this chapter.



```

Rating period:      ERE_PAM_Example_Tree
Rating period file:
Period start time:  Fri Jan 01 00:00:00 CET 9999
Current time:       Tue Jul 07 12:26:37 CEST 2009

Print out of data set:
-- Full DataSet output --
Start time: 20090707 12:26:37
Current time: 20090707 12:26:37
Service: newService
Values:

-- Output done --

ERE_PAM_Example_Tree
===== [ERE_PAM_Example_Tree]
===== [A new Node]
1? <FALSE> Value NOT found
2? <FALSE> Double.test == FALSE (array_field_IN[0] ([VALUE NOT
===== [A new Node]
===== [A new Node]
===== [A new Node]
1! SET Field 'out_numeric_field' with value '5'
1! Qualifier hit list: 1, 14, 45

```

Figure 94 A Sample Simulation Trace Illustrating an STQ Path

6.12 Saving the Selection Tree

To save a selection tree, do any of the following:

- Click the save button in the main toolbar.
- Right-click on the tree and select **Save** from the context-sensitive menu.

6.12.1 Distribution when Saving a Rating Period

When saving the Selection Tree by right - click in the tree, an option of saving the tree to the associated ERE (where the user is working on the tree), or saving it to several ERE instances, appears. For more information regarding the Distribution Wizard, see Section 9.3 on page 206.

6.12.2 Difference Detection Tool when Saving a Rating Period

If work is under progress with several instances of the same Rating Period at the same time, the Difference Detection Tool appears when a save is made. That gives the possibility of comparing the differences between the two trees. This is especially useful if two persons, unwitting, are working with the same tree at the same time. For more information about the Difference Detection Tool, see Section 9.4 on page 208.

6.13 Using the Bookmark Function

The Bookmark function allows the marking of nodes, conditions and modifiers in a selection tree. This makes it easier to find certain elements in a large tree. Bookmarks are added and managed on the **Rating Period** window, using the **Show/hide bookmarks** and **Add a bookmark** icons that appear in the toolbar.



Figure 95 Bookmark icons- Show/hide bookmarks and Add a bookmark

Any node, condition or modifier in the selection tree may be chosen and bookmarked by highlighting the item and selecting the **Add a bookmark** icon. The bookmark created will by default have the same name as the selection tree item, although it is possible to change the name afterwards.

Selecting the **Show/hide bookmarks** icon adds an extra field to the Rating Period window, to the left of the selection tree. A list of all of the bookmarks created for the rating period in question appears in this field.

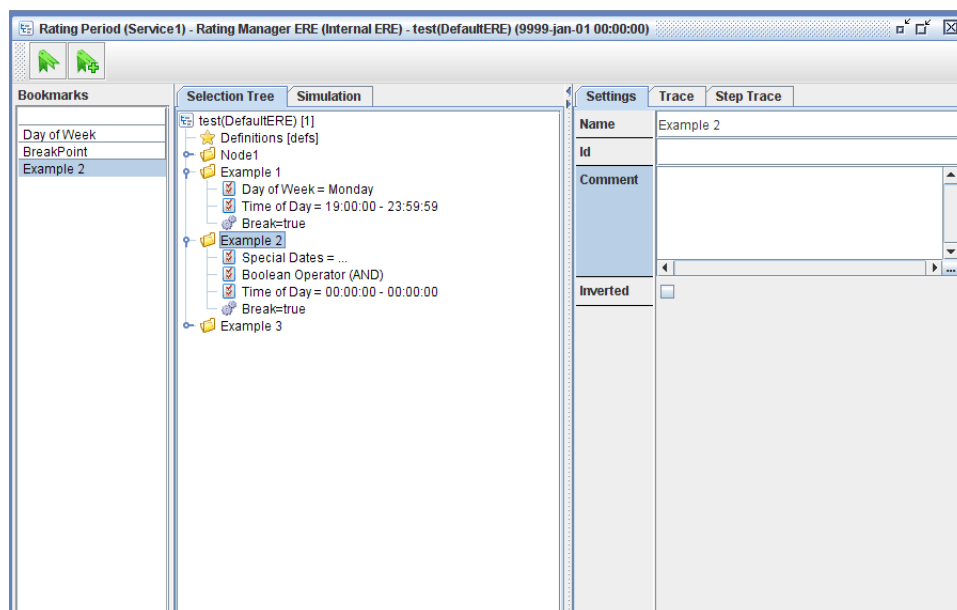


Figure 96 Navigator Window with Bookmarks Field Shown

The bookmarks are path specific, where the full path to the selection tree item bookmarked may be seen by moving the cursor to the bookmark name in the list. Duplicate paths will cause invalid bookmarks. Adding a selection tree path that is identical to another will cause existing bookmarks to be invalid, and make it impossible to add new bookmarks. An error message is shown if an item with a duplicate path and name is chosen as a bookmark.

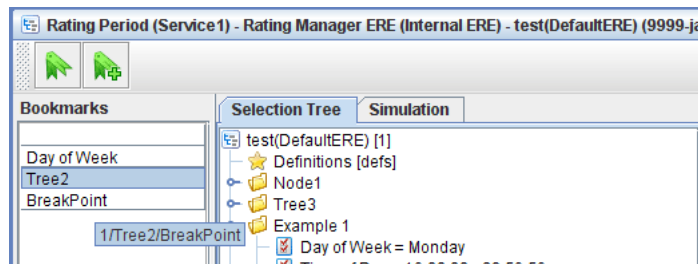


Figure 97 Navigator Window - Path to the Selected Bookmarked Item

Right-clicking on any bookmark in the field will bring up a context sensitive menu, where it is possible to rename or delete the bookmark, or to go to the bookmarked item in the selection tree.

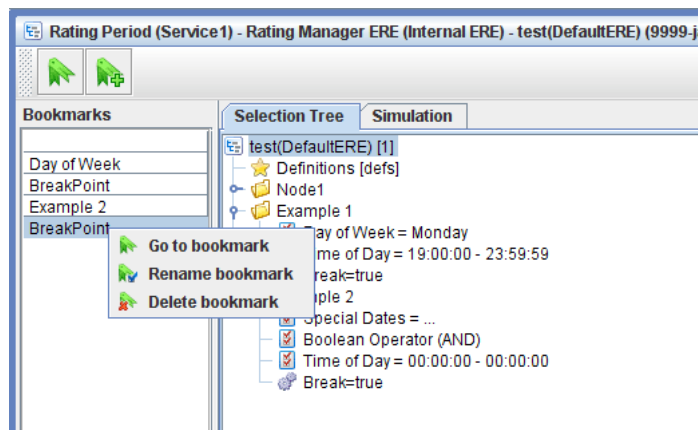


Figure 98 Bookmarks Dialog

Removing a node without deleting the bookmarks contained in that node will lead to invalid bookmarks. Removing an node, condition or modifier with an associated bookmark will also lead to an invalid bookmark. These appear in red text in the **Bookmarks** field, and are marked as invalid. The bookmarks should then be deleted manually to avoid an error message.

Renaming a bookmarked item, or a node which contains bookmarked items will not lead to invalid bookmarks, as the path to the bookmark is updated as the item is renamed. However, renaming the node or item so that the path is identical to another item will lead to invalid bookmarks. These appear in red text in the **Bookmarks** field, and are marked as invalid. One of the selection tree item should then be renamed manually to avoid this state.

The following example describes incorrectly defined bookmarks.

Example: Figure 99 shows two incorrectly defined bookmarks. The nodes named Example2 have the same name and the same path, and are so marked as invalid. The conditions Breakpoint similarly share a name and path, and so are invalid.

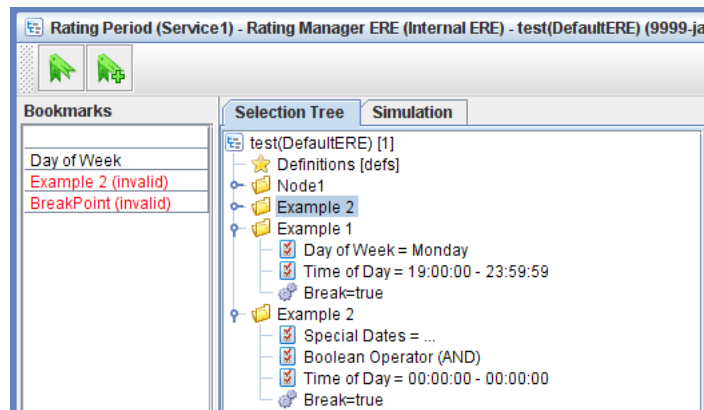


Figure 99 An Example of Invalid Bookmarks

6.14 Using the Find Function

The find function is used to find selection tree items quickly, which is useful for finding certain elements in large trees without the need of browsing through the whole tree. The function is accessed by using the **Find** option in the Selection Tree Editor context-sensitive menu. It is also possible to access it by pressing **CTRL+F**.

For more information on the find function in the Selection Tree Editor, see Section 9.5 on page 211.



7 Testing and Simulating the Selection Tree

A selection tree can be tested in the RMA Simulator to check that it executes as intended. It is a good strategy to test a selection tree every time a change has been made to it before saving. That way, the user can be sure that there are no errors introduced to the selection tree with the latest change.

A selection tree is tested in the simulator by executing it, as it would be done in the ERE environment, but with the users choice of values on the input parameters. Together, the parameters for one simulation make up a simulation case. For the tests to be reliable, they are required to consist of large and well thought out sets of simulation cases. If the tests do not cover all possibilities that the selection tree can be put through when used live, there may exist unwanted behavior in the tree.

The simulator offers the user functionality for creating, managing and using large sets of simulation cases.

Choose the **Simulation** tab in the upper left corner of the selection tree editor to access the simulator.

Depending on the service on which the opened selection tree is built, the Simulator GUI looks similar to the example in Figure 100.

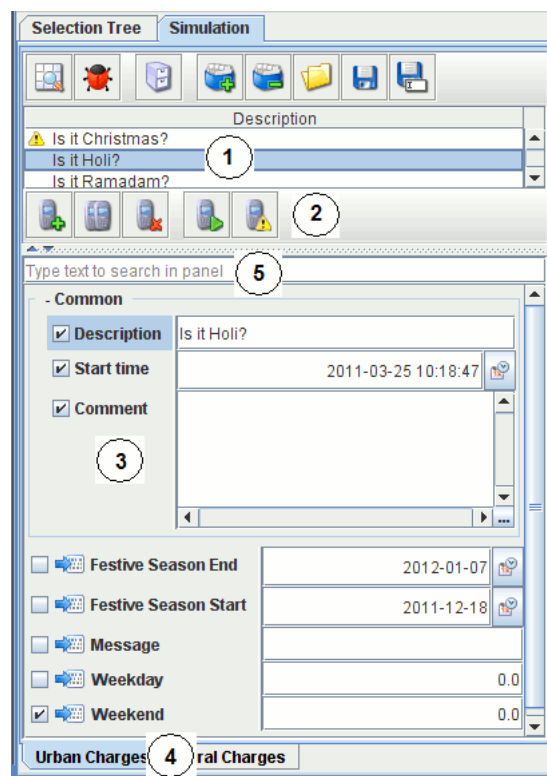


Figure 100 An Example of the Simulator GUI



The components of the Simulator GUI are listed in Table 58.

Table 58 Components of the Simulator GUI

	Components	Description
1	Simulation Table	Gives an overview of the cases in a simulation set, see Figure 100.
2	Simulation Toolbar	Allows actions to be performed on the cases selected in the simulation table, see Table 59.
3	Simulation Data	Shows a more detailed view of a selected case and allows editing of a case.
4	Simulation Set	Stores a set of simulation cases.
5	Simulation Quick Search	Filters the parameters shown in the simulation data based on the parameter name. The filter is case insensitive and hides those parameter names that do not contain the current text as a substring.
6	Simulation Output	Shows the output in different forms from a simulation of a case, see Figure 106 and Figure 107.

7.1 Simulation Case Management

When opening the Simulator, no simulation cases are shown. To create a new simulation case, select the **Adds a new case to the current set** button or the panel stating **Click to create a new case**. The newly created case is then automatically selected in the simulation table, which means that the details of the case are those shown in the simulation data panel.

Columns that describe the cases, for example, their names. It is possible to show additional columns in the table by right-clicking on the column header **Description**. This brings up a popup menu, as seen in Figure 101, listing all the available simulation fields defined in the service. Selecting one of them shows a column in the table with the values of that field from all the current cases.

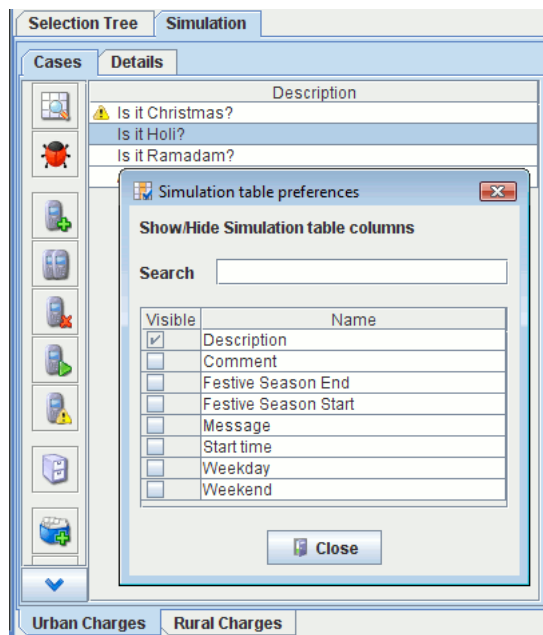


Figure 101 Simulation Table Header











The simulation toolbar holds actions that can be performed on the simulation cases that are selected in the simulation table. All actions listed in Table 59 are also accessible through a popup menu which can be opened by clicking the right mouse button in the table.

Table 59 Simulator Actions

Icon	Description	Hot key
	Switches layout of the simulator. As default, the split view is used. The left panel of the simulator is split between the simulation table and the simulation data details. The simulation toolbar is divided and shown both over and under the table, as in Figure 100. In the maximized view, the table and the data details are both given a separate tab. The toolbar is shown in full to the left of the tabs, as in Figure 101.	Alt+V
	Sets the simulation in debug mode. The behavior of this action depends on the solution that integrates ERE.	
	Creates a new simulation case with default values and adds it to the current simulation set	Ctrl+N



Table 59 Simulator Actions

Icon	Description	Hot key
	Creates one or more new simulation cases based on the ones the user has chosen. First, the simulation cases that should be cloned are selected in the simulation table. When the Clone button is pressed, one new case for every selected is created and added to the current simulation set. Every new simulation case is filled with the values from one of the original cases.	Ctrl+Shift+C
	Deletes the selected case from the set. Deletes the selected simulation cases from the current simulation set. A confirmation dialog is shown to the user before deletion.	Delete
	Executes the selected cases.	Ctrl+Enter
	Toggles between showing all cases in the set and only showing the failed ones.	Ctrl+Alt+F
	Creates a new simulation set.	Ctrl+Shift+N
	Closes the selected simulation set.	Alt+Q
	Switches simulation data source between file and database.	Alt+D
	Opens a simulation set.	Ctrl+O
	Saves the current simulation set.	Ctrl+S
	Saves the current set with another filename.	Ctrl+Shift+S

Note: The Simulator supports storage of simulation cases in a database. This is optional and the appearance of the simulation toolbar is dependent on whether the default file view or the database view is chosen.

By clicking a case in the table, the data parameters for that case are shown.

7.2 Simulation Input Data

What is shown in the simulation input data panel depends on the simulation case that is selected in the simulation table. It is possible to select more than one case



at a time, see Figure 102. If only one simulation case is selected, the input data for that case is shown and made available for editing by the user. If more than one simulation case is selected, it is possible to make changes on all those cases at once. The data panel that is displayed in the Simulator when several simulation cases are selected only contain default values and does not reflect the actual values that the selected simulation cases are holding.

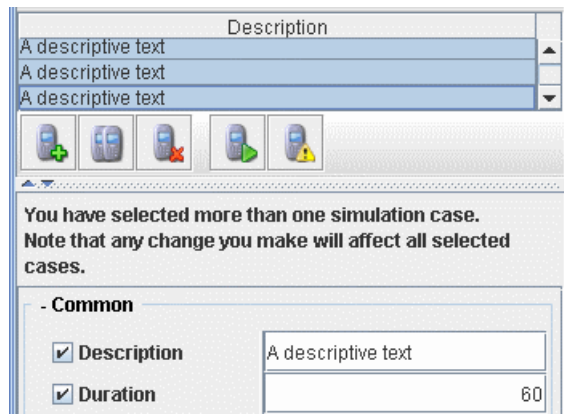


Figure 102 More than One Case Selected

This section describes different input data and input editors in the simulation.

7.2.1 Mandatory in the Simulation

For all configuration fields it is possible to choose if it should be considered or not in the simulation. This is done by selecting or clearing the check box at the left of each field.

- Checked - Use the field in the simulation
- Cleared - Do not use the field in the simulation

Note: If a field in the service definition is indicated as mandatory, the corresponding check box in the main input configuration window is selected as default, and if the field is optional the check box is cleared by default.

The check boxes for the detailed expected output window are cleared by default.

7.2.2 In and Out Fields

A configuration field is either a common field, an in- or outfield, or a validation field. The common fields are dependant of which Simulation Factory that is configured in the Service Definition editor, in other words, they are common to all the service definitions that choose to use a certain simulation factory.

The Figure 103 shows in-, out-, and validation fields in the simulation.

<input type="checkbox"/> Number 1	
<input type="checkbox"/> Number 2	
<input type="checkbox"/> Number 3	

Figure 103 Input, Output, and Validation Fields

In-parameters are considered as input for the simulation, the check box must also be checked for the field to be considered and out-parameters are considered as expected values after a simulation (the check box must also be checked for the field to be considered). If the value, that an outfield has been assigned during the simulation, does not correspond with the expected value that the user has entered, the simulation is considered as failed. Validation fields work the same way as outfields, with the exception that validation fields cannot be modified by the user in the Selection Trees through modifiers.

7.2.3

Grouping

Through the service definition, it is possible to divide fields into different groups, to give the possibility of showing only the groups of parameters that are interesting at the moment. This grouping may affect the order of the data fields in the simulation panel. Therefore, it may not have the same order of the fields as in this example, see Figure 104.

Note: When connecting to an external ERE, the service definition editor is most often not available to the user.

- Common	
<input checked="" type="checkbox"/> Description	A descriptive text
<input checked="" type="checkbox"/> Duration	60
<input checked="" type="checkbox"/> Max cost	100.0
<input checked="" type="checkbox"/> Start time	2007-10-23 12:55:00
<input type="checkbox"/> Expected duration	0
<input type="checkbox"/> Expected cost	0.0
<input checked="" type="checkbox"/> Comment	
+ Tree Defined Fields	

Figure 104 Simulation Panel Showing an Open and a Closed Group

For more information on editing array groups, see Section 6.7.5 on page 90.

For more information on editing map groups, see Section 6.7.6 on page 92.



7.2.4 Parameters Always Available in a Call Case

The parameters described in Table 60, are available in the common group of a call case. Depending on the Service definition that is being used by the Selection Tree which is being edited, this differs. The table lists the fields that come with using the default settings when creating a service definition in RMA.

Table 60 Parameters in the Common Group

Parameter	Function
Description	Possibility of submitting a text that acts as a description of the call case.
Duration	How long the simulation session should be executed, measured in seconds. The duration affects for how long the selection tree should be executed.
Max cost	How much the Max cost for this call case should be. Default Max cost is 100. If the Max cost of the call case is exceeded in simulation, the call case is seen as failed. The trace shows a similar output: Differences found: Max Cost 5.0 NOT enough money for service. Max cost was 2500.0000
Start time	In Start time the user can submit what time the simulation should have as start time. Default is the current date and time.
Expected duration	How long the expected duration is, measured in seconds. If the expected duration is exceeded, the call case is considered as failed.
Expected cost	How much the expected cost is. If the expected cost is exceeded the call case is considered as failed.
Comment	It is possible to add a comment for the call case. This gives a further possibility to explain the purpose of the call case. Any keyboard characters are allowed.

7.2.4.1 Data Fields

The data fields created in the Service Definition are visible among the parameters in each call case. Depending on if they were grouped into different groups or not in

the Service Definition they can appear in one group or more here. The data fields that are not grouped are listed after the common group.

7.2.4.2 Parameters in the Tree Defined Fields Group

TDFs in the tree are visible both in the IN and OUT groups of the Simulation panel. Parameters and their functions are described in Table 61.

Table 61 Parameters in the Tree Defined Fields Group


Parameter	Function
IN	<p>The purpose of this parameter is to simulate the fields that are sent to the tree by a previously executed tree. This is used when simulating a selection tree, without having to simulate a tree that is calling it, for example, through the Go to modifier.</p> <p>The value that should be received in the IN field for each TDF can be submitted. The check box must also be selected for the value to be considered in simulation.</p> <p>Input is restricted to the boundaries of the data type and possible range that the specific TDF has.</p> <p>Default is no value and not mandatory (check box cleared).</p>
OUT	<p>Since it is allowed to modify TDFs, the value they have been assigned in the simulation can be validated in the OUT group.</p> <p>Expected values can be submitted for each TDF. The check box must also be selected for the value to be considered in simulation.</p> <p>Input is restricted to the boundaries of the data type that the specific TDF has.</p> <p>Default is no value and not mandatory (check box cleared).</p>

Note: If no TDFs have been created, these groups are empty.

7.2.5 Validation of Input

All input is validated according to what constraints the field has for the data type itself plus other input restrictions, such as range. Faulty input is marked by marking the field with a red color. Tables mark the faulty row with a red frame. The call case is also marked as red in the call case table. Different input editors have diverse ways of showing faulty input. It is not possible to simulate an execution of a simulation case containing faulty values.

7.3 Simulation Procedure

The Execute case action, icon , starts a simulation with the selected case as input. The simulation generates output in the form of a trace printout as well as a trace tree. If multiple cases are selected when simulating, one simulation is executed for each of the selected cases. In this case, no trace printout or trace tree



is available, but a summary of how many cases that were successful is given. To get the detailed simulation output, the cases have to be executed one at a time.

When a simulation case has been executed, it is marked with its simulation status (failure or success) in the table. A green selected icon represents success and a yellow warning triangle represents failure, see Figure 105.



Description	
	A descriptive text
	A descriptive text

Figure 105 The Icons for Execution Succeeded or Failed

Success or failure of simulation depends on whether the expected values entered in the out fields and validation fields match the actual values set during simulation.

The Switch between all or failed action, icon , filters out all cases that either have succeeded in simulation or that has not been simulated at all, leaving only the failed ones. This gives the user a better overview of the cases that requires further investigation. Pressing the button again shows all the cases in the simulation set.

7.4 Simulation Output

The simulation output is shown in a detailed trace and as a tree overview of what have been executed in the tree. The simulation output displays what happens when the rating logic in the selection tree is executed.

7.4.1 Step Trace

The step trace reflects how the selection tree has been executed and it enables easy identification of nodes, conditions, and modifiers that have been executed. It is possible to see details about the execution of each condition and modifier by selecting it in the tree. Faded object has not been executed. It is possible to hide non-executed branches and child objects by enabling the **Show only executed branches in step trace** setting in **RMA Settings > Selection Tree Editor**, see Section 4.2 on page 19.

An example of a print out is shown in Figure 106. For description of the different parts of the trace tree, see Table 62.

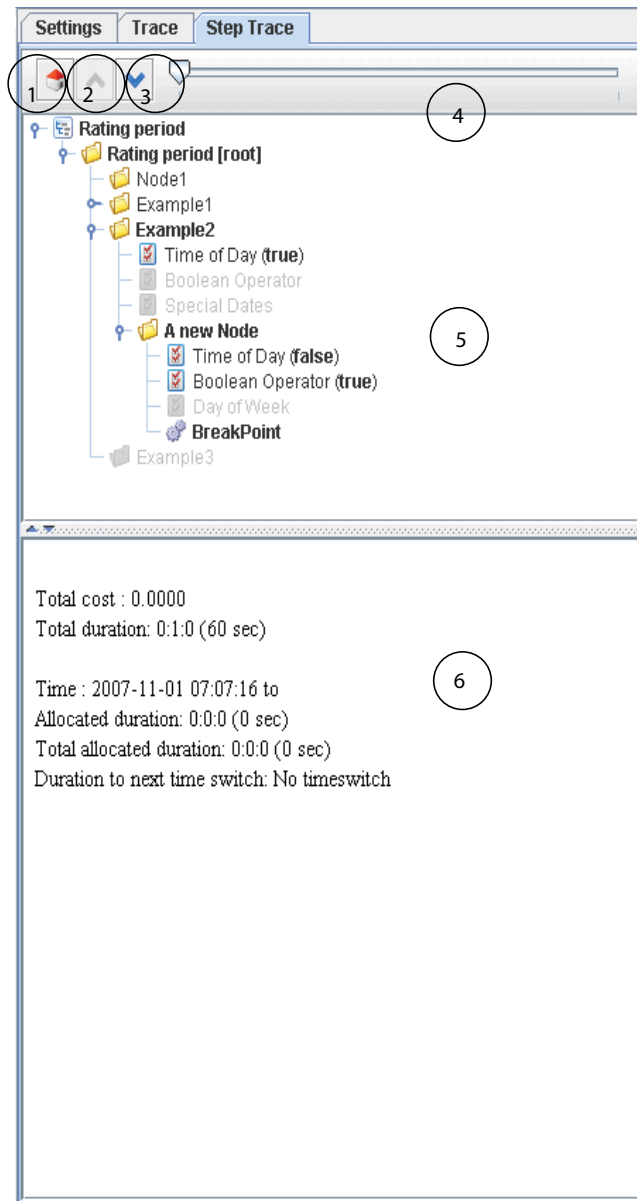


Figure 106 Example of a Print Out

7.4.1.1 Simulation Toolbar

Table 62 The Simulation Toolbar

Section	Component	Description
1	Show summary	Printout regarding the whole execution. Generally shows when the execution started, ended, and also duration and cost.



Table 62 The Simulation Toolbar

Section	Component	Description
2	Go back to previously executed modifier	Steps up to the prior modifier in the selection tree that has been executed.
3	Step forward to the next executed modifier	Steps down to the next modifier in the selection tree that has been executed.
4	Slider	The slider enables stepping between all executed modifiers by dragging it to the left or right. The first executed modifier is present to the far left. The last executed modifier is present to the far right.
5	Tree	Visualization of the executed tree. Objects that have not been executed are faded.
6	Trace Printout	Specific printout for each condition or modifier. Printout regarding the whole execution. Generally shows when the execution started, ended and also duration and cost.

7.4.2 Trace

The trace output shows a detailed output of the parameters after a simulation, see Figure 107.



```
===== [A descriptive text] =====
Service: Service
Service provider: Service Provider
Rating plan: Rating Plan
Rating period: Rating period
Rating period file:
Period start time: Fri Jan 01 00:00:00 CET 9999
Current time: Thu Nov 01 07:07:16 CET 2007

Print out of data set:
-- Full DataSet output --
Start time: 20071101 07:07:16
Current time: 20071101 07:07:16
Service: Service
Values:
- groupA

-- Output done --
Max cost: 100.000000
Max duration: 60
Current cost: 0.000000

Rating period
===== [Rating period]
===== [Model]
===== [Example1]
1? <FALSE> The day of week (Thu) is NOT between Mon and Tue
2? <FALSE> Time (2007-11-01 07:07:16 ) is NOT between 01:00:00 and 05:00:00
===== [Example2]
1? <TRUE> Time (2007-11-01 07:07:16 ) IS between 06:00:00 and 12:00:00
===== [A new Node]
1? <FALSE> Time (2007-11-01 07:07:16 ) is NOT between 01:00:00 and 05:00:00
2? <TRUE> Using AND on the result from all children.
1! Stopped tree execution

Parameters that were changed:

Total cost : 0.0000
Total duration: 0:1:0 (60 sec)

Time : 2007-11-01 07:07:16 to
Allocated duration: 0:0:0 (0 sec)
Total allocated duration: 0:0:0 (0 sec)
Duration to next time switch: No timeswitch
```

Figure 107 Trace Output

The Table 63 explains the trace output figure above.



Table 63 Explanation of the Figure Above


Simulation output section	Description
1	<p>Information regarding the selection of the rating period used in the trace.</p> <ul style="list-style-type: none"> • Service – shows what Service this Rating Period has. • Service Provider – shows what Service Provider this Rating Period has. • Rating Plan – shows the Rating Plan this Rating Period belongs to. • Rating Period – shows the name of the Rating period. • Rating Period File – if the Rating Period has been saved to file its path and the name of the file is specified here. • Period Start Time – shows the start time for this Rating Period. • Current Time – shows the time when the simulation executed the tree.
2	<p>The Full DataSet Output always shows:</p> <ul style="list-style-type: none"> • Start time – the start time for this execution. • Current time – the present time (at printout). • Service – the service that this Rating Period is connected to. <p>It shows all the data fields that were set as mandatory and therefore were considered in the simulation. It also shows values they have received through the execution (if applicable).</p>
3	Full representation of the tree that has been executed.
4	Parameters that have been changed, which means parameters that have received another value during execution.
5	<p>Printout regarding the whole execution.</p> <p>Total cost – shows the sum cost of this execution.</p> <p>Total duration – shows for how long the call case executed. This is dependent on what values were submitted in the Duration parameter.</p> <p>Start time – at what time this execution begun.</p> <p>End time – when execution ended</p> <p>Allocated duration, Total allocated duration, or Duration to time switch.</p>

7.5 Simulation Sets

Simulation Sets are collections of simulation cases. By default, when opening a selection tree editor, a set is created. It is possible to create new simulation sets that are empty or that contains simulation cases from a preexisting place, that is from file or from a database.

Simulation Sets are a way of keeping collections of simulation sets apart, but at the same time making it possible to have them open in the same selection tree editor. The Simulator has no restrictions of how many Simulation sets that can be open at the same time.

A simulation set can be saved to either file or database regardless from where it was opened. This is described further under Section 7.6 on page 127.

The Add simulation set, icon , opens the Create Simulation Set dialog shown in Figure 108. This dialog lets the user decide with what the newly created set should be filled. The default option is to create an empty set which is added to the already existing sets at the bottom left corner of the simulator. After that, it is free to add new cases to it or to load cases from a data source.

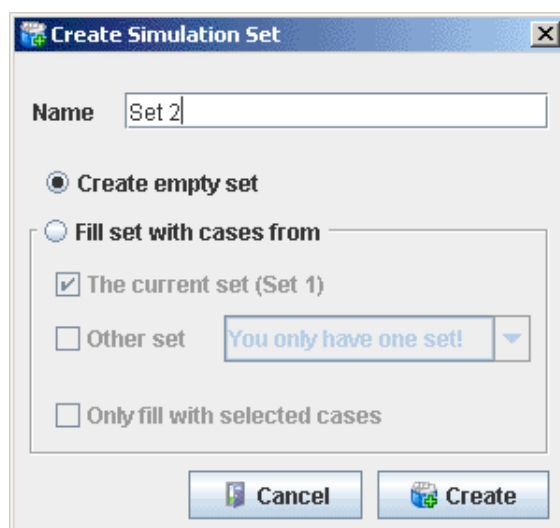



Figure 108 Simulation Set Creation Dialog

A simulation set, that has already been created, is cloned by choosing to fill the new set with cases from the currently viewed one. But if the goal only is to extract a part of an open simulation set, first select the cases that are to be extracted in the simulation table of the currently viewed set. Then Press the **Add Set** button and choose to add the selected cases from the current set to the new set.

It is also possible to merge two open simulation sets in the Create Set dialog. The new set then consists of all the cases from two arbitrary chosen sets, or if desired only the ones that have been selected in the simulation table.




The Close simulation set action, icon , closes the currently viewed set in the simulator. If there are unsaved changes, a popup appears with a question if the set should be saved before it is closed or not. The currently selected data storage is used. It is possible to press **Cancel** to get back to the set and change it before saving.

7.6 Simulation Case Storage


The Simulator makes it possible to open and save sets independent of each other to either file or database. Saving to file is easier and does not require setting up a database, but database storage is preferred when the simulation set is growing large. If there are more than one user working with the Selection Trees, it is a good idea to maintain a central simulation case database accessible for all.



The Switch data source action, icon , switches the simulation toolbar between the file storage view and the database storage view. There is no actual change in storage for the current set done at this time. It is a visual action to limit the options. To save the set to a new location, choose the **Save** option.


7.6.1 File Storage




Open simulation set action, icon , enables browsing for a simulation file on the computer or on a network drive to open. Should there be differences between the opened set and the Service definition, for example, missing Simulation Fields, information about this shows and these fields are removed or added.

If the currently viewed simulation set is not empty when opening, the user is prompted to go ahead before the set is emptied and filled with the other cases.




The Save simulation set action, icon , saves the current simulation set to a location specified by the user. If the set has been saved once before during the editing session, the same location is used next time. Should it not have been saved before, a browse dialog appears so the location where to save it can be specified.



The Save simulation set as action, icon , saves the current simulation set to a specified location through a browse dialog. If the set has been saved once before during the editing session, that location is replaced by the newly selected one as the new default location for the save action.

7.6.2 Database Storage

The Configure simulation **New Database Connection** action, icon , is located in the **Util** menu of RMA. It opens the **Simulation Database Configuration** dialog shown in Section 7.6.2 on page 127. To work with a simulation database, a database connection must be set up. The Simulation Database Configuration dialog enables adding, editing or removing database connections.

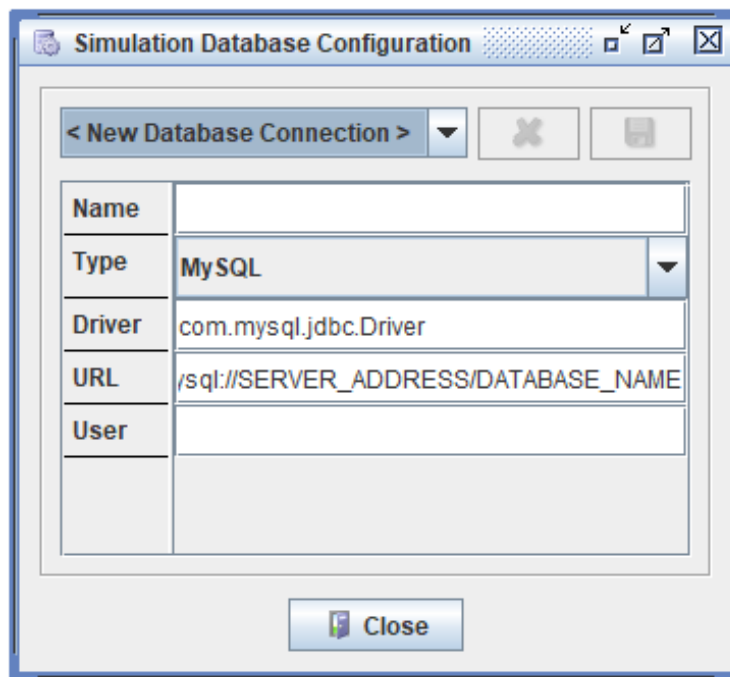


Figure 109 Simulation Database Configuration Dialog

RMA is not responsible for configuring the database, this must be done manually. The settings in the dialog should match the ones that have been set up in the database server that the user wants to connect to. The fields that must be filled in are listed in Table 64.


Table 64 Simulation Database Configuration Settings

Field	Description
Name	A name of the database.
Type	The type of the database. (Chosen from a list of database types supported by RMA).
Driver	Java class path to the Java Database Connectivity (JDBC) driver required by RMA to connect to the database. See Section 7.6.2.1 on page 130 for more information on where the driver is located.



Table 64 Simulation Database Configuration Settings

Field	Description
URL	An address to the database. It should be written on the form: jdbc:TYPE://ADDRESS/NAME where TYPE is replaced by the database type, ADDRESS is replaced with the IP address where the database is residing, and NAME is replaced with the actual name of the database.
User	The user name required to login to the database.

The **Open simulation set** action, icon , opens a simulation set from a database similar to the file storage variant. The selection of from which database the set should be loaded is done through the **Simulation Database Connection** dialog, shown in Figure 110.

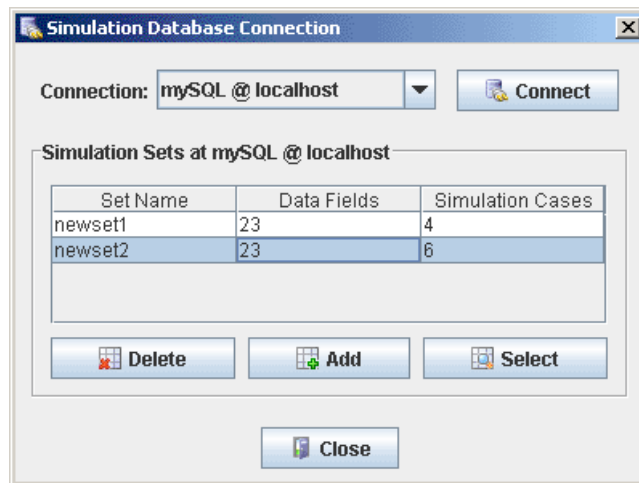


Figure 110 Simulation Database Connection Dialog


1. Choose a predefined connection from the **Connection** drop-down.
2. Press **Connect**. This lists all available sets on that database.
3. If the database connection does not exist, a **Database Connection Password** dialog box will appear as shown in the following figure:




Database Connection Password

4. Enter the database connection password and click on **OK** button. Details for every set are shown next to the set name.

It is possible to select one simulation set by pressing **Select**.

The **Save simulation set** action, icon , works similar to the file storage version, but instead of a browse dialog, the Simulation Database Connection dialog is shown.

Save simulation set as action, icon , works similar to the file storage version, but instead of a browse dialog, the **Simulation Database Connection** dialog is shown.

7.6.2.1 Location of the Simulation Database Driver

The location of the JDBC driver required by RMA to connect to the database depends on if the ERE is started as a standalone application, or is started from a web browser.


When ERE is started as a standalone application, the Java Archive (JAR) file which contains the driver should be located in the RMA lib folder on the computer from where RMA is started.

When ERE is started in a web browser, the JAR file should be installed in one of the Java extension directories. The location of Java extension directories can be identified from within RMA. On the main menu bar, select **Help** and then **About**. In the dialog that appears, switch to the **Runtime Properties** tab. Under the Java Properties heading, locate the row with the property named `java.ext.dirs`. In the Value column is a list of directories in which the Java Runtime Environment looks for extension libraries. Install the JAR file in one of these directories, and then restart RMA."

7.6.3 SQL Tool

When storing simulation cases in a database it is possible to create large amounts of cases and save for future use. A large set of test cases might be hard to outline and maintain. Therefore the Simulator comes with a SQL Tool that enables searching in a set of simulation cases stored in a database. The tool can be configured to filter out simulation cases by defining what their simulation fields contain. The filtered set of simulation cases can then be extracted to a new simulation set in the Simulator.

7.6.3.1 Working with the SQL Tool

The SQL Tool is found in the Simulator when in Database storage view. The Storage views are explained further in Section 7.6 on page 127. To start the tool, press the **SQL Tool** icon . When started, the user is asked to choose a saved Simulation set to base the search on, see Figure 110. When a set is chosen the



SQL Tool window is opened. The main window is explained in Figure 111 and Table 65 below.

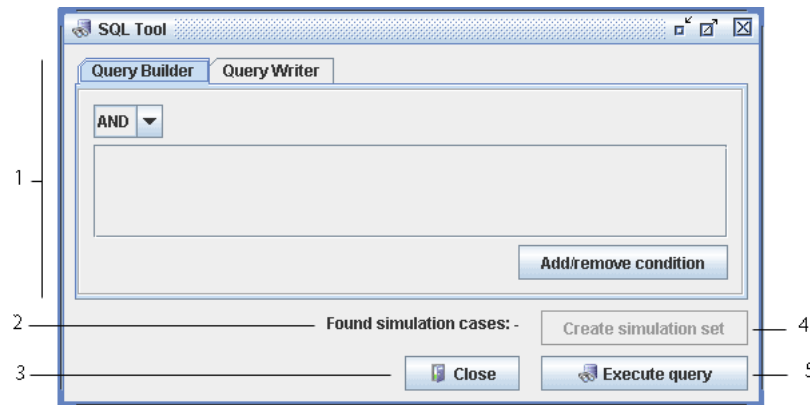


Figure 111 The SQL Tool

Table 65 Component Description

Section	Component	Description
1	Tabbed panel	The function tabs. Available tabs are Query builder and Query writer.
2	Label	Presents the number of found simulation cases.
3	Close-button	Closes the SQL tool.
4	Create simulation set-button	Creates a simulation set in the Simulator and inserts the simulation cases in the result from the last executed query.
5	Execute-button	Executes the currently configured query.

The SQL Tool consists of a main window with two tabs: Query Builder and Query Writer. The tabs represent the two functions of the tool:

The Query Builder

The Query builder enables composing any number of search criteria, here called conditions, to use when fetching test cases from the database. A condition is always based on the content of one of the data fields in the simulation cases.

Note: It is not possible to use a data field that belongs to a context in the search.

The Query builder tab is selected when the SQL Tool is opened. Initially there are no conditions configured. Figure 112 and Table 66 explains the basic parts of the Query builder.

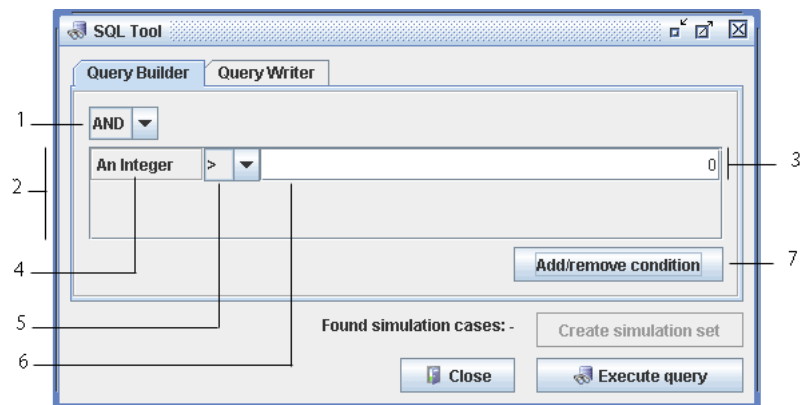


Figure 112 The Query Builder

Table 66 Component Description

Section	Component	Description
1	Combo box	Combo box for choosing if the conditions in the list should be set up with “OR” or “AND” in the query.
2	List	The list with the configured conditions.
3	List item	A configured condition.
4	Label	The name of the data field of this condition.
5	Operator-combo box	Combo box with the available operators for this condition.
6	Depends on the data field connected to the condition	The parameter input component for this condition.
7	Add/remove condition button	Opens the Selection tool for adding and removing conditions.

To add a condition, press the button **Add/remove condition**. The selection tool, see Figure 113, is shown. In this tool the available data fields are shown in a table. Click to check the check box to the left of the wanted data field to add it to the condition list. To remove a data field, click to clear the check box to the left of it. An additional feature in the selection tool is the **Search** text field that is used to filter the data fields in the table. By typing a part of the name of a wanted data field, the table is updated to show only the data fields that match the text string in the **Search** text field.

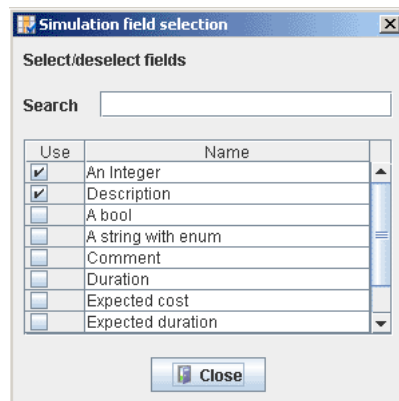


Figure 113 Selection Tool

The condition list in the SQL Tool is concurrently updated with the choices made in the selection tool.

The chosen data fields are now shown in the Query builder window as rows in the condition list. Conditions can be configured to represent the wanted search criteria. This is done by choosing an operator and a parameter. Available operators depend on the configuration of the chosen data field. Table 67 below explains the operators.

Table 67 Available Operators

Field data type	Available operators
Boolean, Date, Time, OctetString	Equals – an exact match with the parameter.
String, BcdString	<p>Begins with – matching the string with the beginning of the data field value.</p> <p>End with – matching the string with the end of the data field value.</p> <p>Contains – matching any occurrence of the string.</p> <p>Exact match – an exact match with the string.</p>
Numeric data type	<p>>, bigger than the parameter.</p> <p><, less than the parameter.</p> <p>>=, bigger than or equals to the parameter.</p> <p><=, less than or equals to the parameter.</p> <p>=, equals to the parameter.</p> <p>!=, not equals to the parameter.</p>

The format of the parameter is also dependent on the chosen data field. For example, if a boolean data field is chosen, the parameter component is shown as

a check box, while a string data field is shown as a text field. Any input restrictions configured for the data field, also affects the parameter component behavior.

If more than one condition is configured, it must be decided how they should be treated when inserted in the query. The options are: ONE of the conditions must be true (OR), or ALL conditions must be true (AND). This is chosen in the combo box in the top left corner of the tool, see Figure 112.

When the configuration is satisfactory, the query can be executed as explained below.

The Query Writer

The Query Writer tab enables the user to write a customized SQL query. There is a text area in which the query is typed in. The button named **Get query from query builder**, copies the SQL query configured in the Query Builder tab to the text area, and the user can then modify the query before execution.

Figure 114 and Table 68 explains the basic parts of the Query writer.

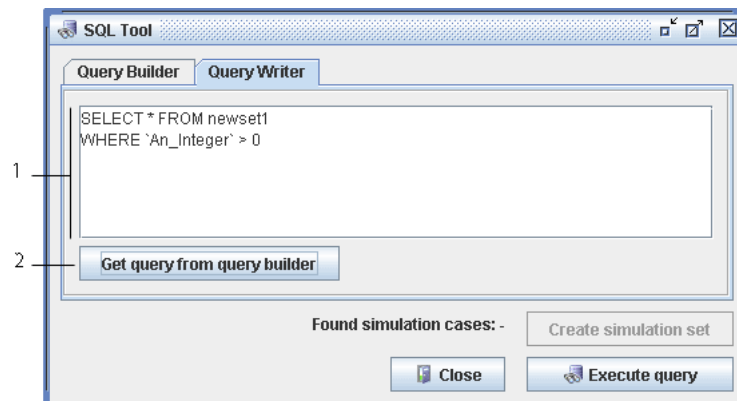


Figure 114 The Query Writer Tab

Table 68 Component Description

Section	Component	Description
1	Text area	Text area where an SQL query can be written.
2	Get query from the button Get query from query builder	Copies the query configured in the Query Builder tab, to the text area.

The only constraint on the query defined in the Query writer is that it must begin with the text string: **SELECT*FROM**. This is for prohibiting an execution of an SQL statement, that harms the simulation cases stored in the database.

When the configuration is satisfactory, the query is executed as explained below.



Executing the query

When the query configuration is satisfactory the statement is executed using the **Execute** button in the lower right corner, see Figure 111.

The number of matching simulation cases are prompted in the bottom right corner. It is always possible to go back and modify the query in the Query builder or the Query writer, and execute the new configuration.

When a number of simulation cases have been found, it is possible to open them in a new simulation set in the Simulator. This is done by pressing the **Create simulation set** button in the bottom right corner of the tool.

7.7 Selection Tree Test Tool

The purpose of the selection tree test tool is to be an aid when testing selection trees. A well maintained collection of tests is useful when regression testing, that is, discovering errors as a result of recent modifications to a selection tree.

The tool is configured through a wizard user interface where it is possible to do the following:

- Affect the number of test possibilities that needs to be tested for every condition.
- Configure means of validating if the generated test cases succeed.
- Join several selection trees through modifiers into a single generation path.
- Decide which values that are going to be used as default values for the generation.
- Name the simulation cases that are produced and the simulation set containing them.

7.7.1 Starting the Test Tool

The test tool is started as in Figure 115 by choosing a modifier in the selection tree editor, and choosing **Generate Simulation Set** (hot key **Ctrl+G**).

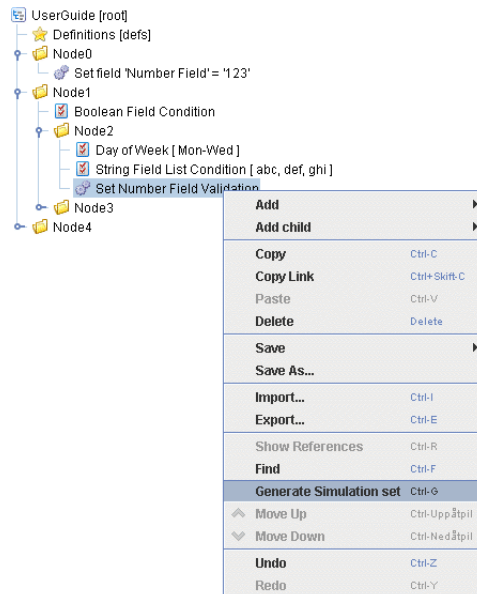


Figure 115 Example: Tree for Generating Simulation Cases.

7.7.2

Configuring Conditions and Modifiers

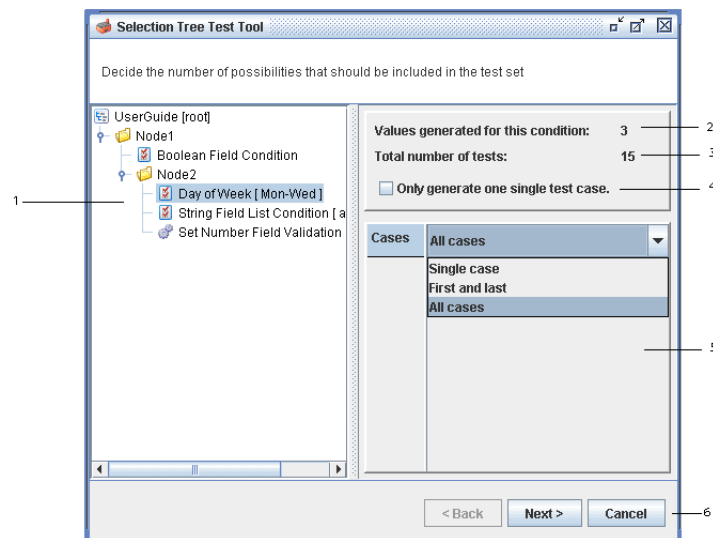


Figure 116 Wizard Page 1 - Condition Settings

The first page of the test tool wizard is explained in Table 69.



Table 69 Wizard Page 1

1	Generation path	The path leading from the root of the selection tree down to the selected modifier. Only the conditions shown here can affect the generation.
2	Selected condition test count	The number of possibilities taken into account when generating test cases where the selected condition evaluates true.
3	Total test count	The total number of test cases that are generated.
4	Test settings override	Overrides the condition-specific test settings and tells the generator only to generate one single test case.
5	Tree element specific settings	Shows the settings for the condition or modifier that is currently selected in the generation path.
6	Navigation panel	Holds buttons for closing the wizard or moving forwards and backwards in the wizard.

The test tool produces simulation cases for all the possible ways of reaching a certain modifier. The possible ways of reaching a modifier are a result of the conditions that lies on the path leading from the root of the tree down to the modifier.

The settings for a Condition determine the number of different ways that the condition can be evaluated true in a simulation. For example, a **DayOfWeek** condition, as seen in Figure 115, set to Monday-Wednesday is true for Monday, Tuesday, and Wednesday. In that example, three different simulation cases are generated for that condition, one for each day.

The selection tree, shown in Figure 115, is used as an example of how to use the test tool wizard.

The wizard allows the user to limit the number of simulation cases generated. This is done in the test settings pane displayed on the right hand of the wizard when selecting one of the conditions. For example, it may not be necessary to generate simulation cases for all the days in the **DayOfWeek** condition or all the strings in the **String Field List** condition. This knowledge lies with the designer of the selection tree and that is why the wizard prompts the user.

As seen in Figure 116, the **DayOfWeek** condition is selected. In the drop-down editor, **All cases** is selected, which eventually result in 3 cases (Mon, Tue, Wed). The counters shown above the settings panel in the wizard, keep track of the number of simulation cases that are generated.

The counters show both the number of cases that is required for the currently selected condition as well as the total number of simulation cases. The user should be aware that the total number of simulation cases is not a sum of the cases generated for each condition in the path. To test all the possible ways of reaching the modifiers, even cases where some of the conditions evaluate false are created.

It is possible for the user to override the condition-specific settings and let the wizard generate one single simulation case. This is done by checking the check box **Only generate one single case**.

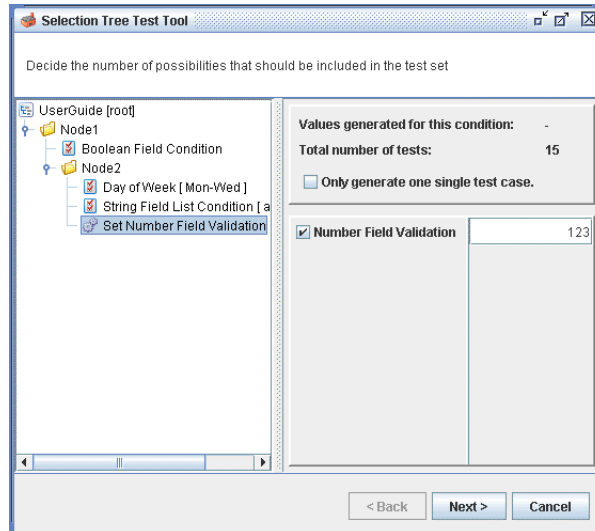


Figure 117 Wizard Page 1 - Modifier Settings

When a modifier is selected, the modifier-specific settings are shown (See Figure 117). Modifier-specific test settings that are marked with a check box are validation settings. Some modifiers do not need any manual configuration. In this example, the purpose of the modifier is to set the field **Number Field Validation** to whatever value the **Number Field** field is set to.

Changes to the data outside of the generation path cannot be taken into account by the test generator and has to be recognized by the user. Therefore in this example, the **Number Field Validation** field should be set to the value that the user expects the field to have when the modifier is reached.



7.7.3 Setting Default Values

Figure 118 Wizard Page 2

The next wizard page, shown in Figure 118, makes it possible to specify the default values that are used as a basis for the test generation. Any values that are entered here may be overwritten by the generator if needed.

If the user knows about any changes to the data done by modifiers outside the generation path, they should be entered here. In our example, the **Number Field Validation** field should be initiated with the value the user believes the field to have at the point of execution where the generation path begins (in this example at Node1).

7.7.4 Naming the Tests

Figure 119 Wizard Page 3

The next wizard page is described in Table 70

Table 70 Wizard Page 3

1	Simulation set name	Name of the resulting simulation set.
2	Auto naming option	Decides if the wizard should auto generate names for the simulation cases based on the nodes and modifiers included in the generation path.
3	Include nodes	Include node names in the auto generated name.
4	Include modifier	Include modifier name in the auto generated name.
5	Manual naming option	Decides if a prefix, entered by the user, should be included in the simulation case names.
6	Prefix	The prefix to be included in the case names.
7	Naming example	An example of a simulation case name that changes depending on the settings made on this wizard page.

The next wizard page, shown in Figure 119, allows the user to affect how the generated simulation cases, and the set that contains them, are named. The names of the nodes as well as the name of the modifier in the generation path are included by the appropriate check boxes. It is also possible to enter an arbitrary text, that is used as a prefix in the simulation case names.

Regardless of user options, a number is always added to the end of the case name to make it unique.

7.7.5 Starting the Generator

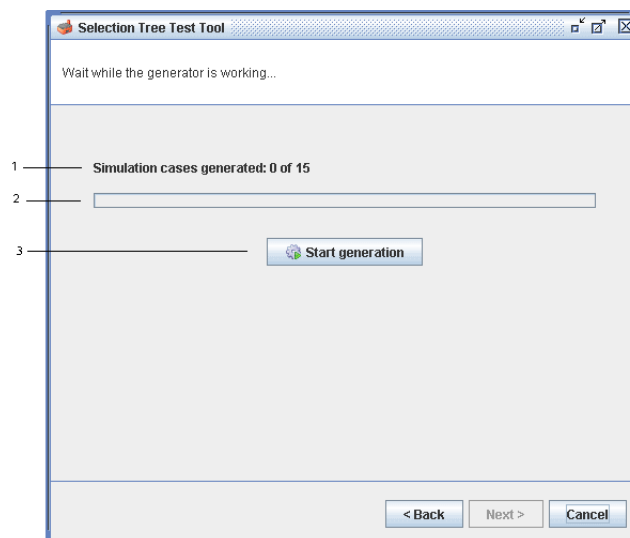


Figure 120 Wizard Page 4

The next wizard page is described in Table 71.



Table 71 Wizard Page 4

1	Test count	Shows how many of the total tests that have been produced.
2	Test progress bar	Shows how many of the total tests that have been produced.
3	Test generator control	Lets the user start, stop or restart the test tool.

Navigating to the next wizard page, seen in Figure 120, and choosing **Start generation** starts the test case generator with the configurations made in pages 1-3. When it is finished, it is possible to move on to the next page. It is also possible to move backwards in the wizard, if the user wants to modify some of the settings made, and start a new test generation.

7.7.6 Test Tool Result

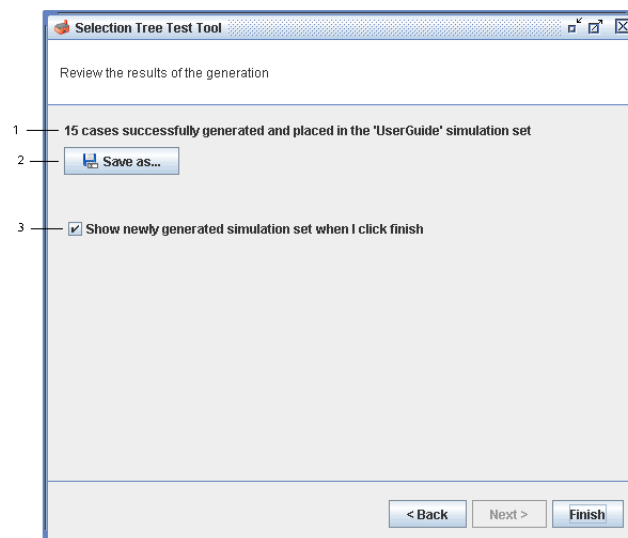


Figure 121 Wizard Page 5

The next wizard page is described in Table 72

Table 72 Wizard Page 5

1	Test result	Shows the properties of the resulting simulation set.
2	Save result set	Possibility to save the result set to file.
3	Open result set	Possibility to open the result set in the simulator panel.

When the test generator has completed successfully, the user has the opportunity at the last wizard page, Figure 121), to save the simulation set containing the cases to a file, and also to inspect the cases in the simulator panel.

7.7.7 Test Tool Characteristics

- The generation path leading to a modifier may contain conditions that are not supported by the test tool. When starting the wizard, the user is notified that unsupported conditions exist in the path. These conditions are ignored by the wizard, that is, they are not able to affect the number of test cases that are produced. It is, however, possible to go ahead with the generation and to modify the produced simulation cases manually later in the simulator panel.
- The user is also notified if several conditions in the generation path are affecting the same field, for example, the **TimeOfDay** condition and the **DayOfWeek** condition both need the Start Time field to be set for them to be evaluated. Such situations are complicated for the test tool and may result in that unexpected test cases are produced, for example, cases that fail or identical cases. Thus, it is highly encouraged to manually inspect and execute the simulation cases generated by the test tool.
- It is possible to generate simulation cases for a generation path that spans across several selection trees, joined using the goto modifier:

Start the wizard by selecting a terminating modifier in the sub-tree, that is, the tree that is being called.

Select the root of the sub tree selected in the wizard. Where the condition or modifier-specific setting are usually shown, the wizard now displays a text stating that it is possible to drop a goto modifier at that location to combine the tree containing the goto modifier and the sub-tree. This can be achieved by opening the selection tree with a goto modifier, and drag a goto modifier from the tree to the wizard.

When the two trees are joined in the wizard, it continues to function as before.

- It is important to know that the tool, depending on the complexity of the selection tree, may not be able to produce a simulation set covering all the possibilities of input data for the tree. It is vital that the user sees the tool not as a complete solution, but as an aid providing a set of simulation cases which is a good basis that the user can continue working with.
- It is not possible to start the test tool from some of the modifiers in RMA. This may be because they are not “terminating” modifiers, that is, the execution of the selection tree does not stop when the modifier is encountered. It may also be because the modifier does not set any validation or output fields.



8 Plug-in Descriptions

A plug-in can be a node, condition, or modifier that is used in the selection tree of a rating period for setting up the logic. The plug-ins that should be available for each rating period are declared in the service connected to it. For further information on how to add plug-ins, see the Section 5.4 on page 51. For general examples on how to set up selection tree logic see section Section 6.10 on page 104.

This chapter contains descriptions of the general ERE plug-ins that can be used to create or edit selection tree structures. The descriptions in this section explain the parameters each plug-in includes and in some cases examples of use.

Depending on the solution that the ERE is integrated in, some of the plug-ins may not be available in a specific selection tree. This depends on what service it is connected to.

8.1 Nodes

A node is a container for a collection of child plug-ins - sub-nodes, conditions and modifiers- which are evaluated in their given order to do a specific task.

8.1.1 Inverted Nodes

A node can be inverted, which means that the node is considered to be true when all conditions are evaluated as false. A node is inverted by selecting the check box **Inverted** in the node settings.

Figure 122 shows the difference between the evaluation of a regular node and an inverted node.

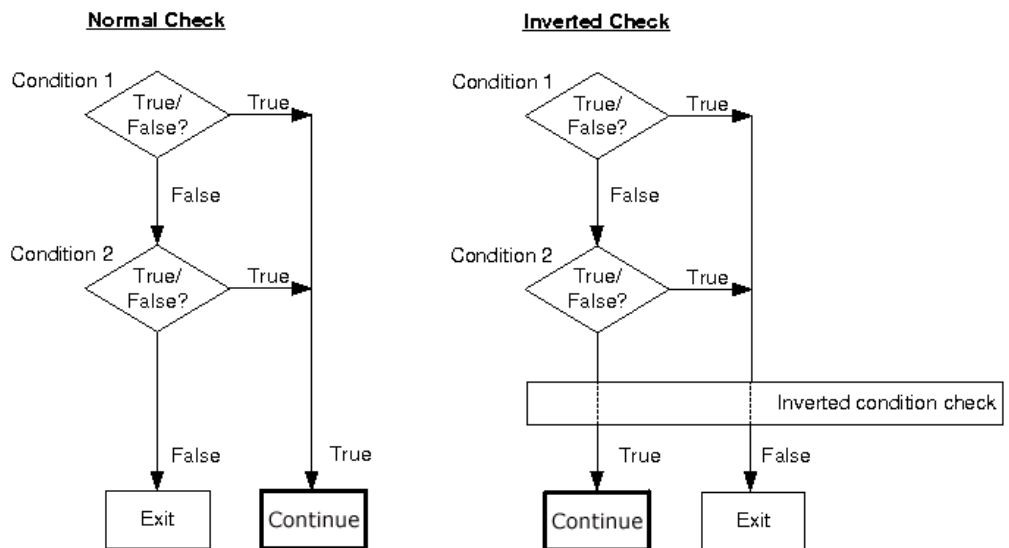


Figure 122 Difference between Normal and Inverted Check

Inverting a node is useful in cases where it is easier to list the exceptional cases than all cases that are valid. Inverting a node does not affect sub-nodes contained in the node, which may in turn be inverted.

In the example below, a **NumberList** condition is used with the numbers 1, 2 and 3 listed. In this example, the desired outcome is that all numbers that are not 1, 2 or 3 should be true and that the node continues to execute. Therefore, the node that the condition is placed in is set as inverted. The inversion does not affect the sub-node Example 1.1 in Figure 123.

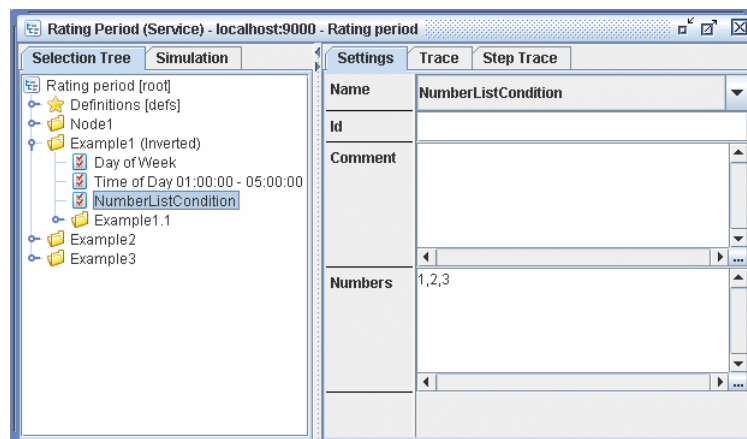


Figure 123 Inverted Node


8.1.2

If/Else Nodes

An If/Else node behaves differently from a regular node. A regular node is stepped through until the final valid modifier is evaluated, and then exits. An If/Else node is



stepped through until a valid modifier is evaluated, and then the node is exited. Any subsequent sub-nodes are not evaluated.

If/Else nodes are marked with OR, , in the **Selection Tree** tab.

All conditions in an If/Else node must be contained in sub-nodes. If a node is defined as If/Else when it is created, only a sub-node may be added as a child. If a regular node is converted to an If/Else node, all conditions and modifiers outside sub-nodes are removed.

An example of how an If/Else node can be used may be found in Section 8.1.2.1 on page 145.

8.1.2.1 Defining an If/Else Node

In the example in Figure 124, if the node is evaluated on a Thursday, the result from an If/Else node is "Soon weekend". A regular node would exit the "Soon weekend" sub-node, continue to evaluate the node and return the result "weekend", regardless of what day of the week it actually is.

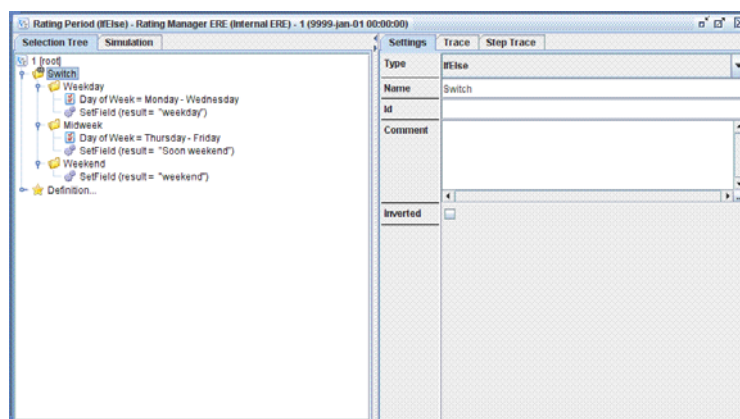


Figure 124 An example of an If/Else Node

8.2 Conditions

This chapter contains descriptions of all available base and generic conditions and their parameters. For an explanation of the condition concept, see Section 2 on page 3.

The class names described in the following tables are partial, and all have `com.ericsson.ere.selectiontree.conditions` as a prefix. For example, Short Condition is fully qualified as `com.ericsson.ere.selectiontree.conditions.number.ShortCondition` as the class and `com.ericsson.ere.selectiontree.conditions.number.ShortConditionProfile` as the profile.

The class names are used when defining conditions in the service editor.



8.2.1 Base Conditions

Base conditions are conditions used in a selection tree, where they can be assigned to a specific field using the service editor, see Section 5.3 on page 34, or using the **FieldSelection** condition, see Section 8.2.3.9 on page 180.

One base condition can be connected to many fields of compatible data types. For example, the Integer condition can be connected to several data fields of the type integer. Each data field is an instance of the Integer base condition.

The base conditions are presented in Table 73, and further described in the sub-chapters. The names in the table are merely suggestions, it is possible to give the instantiated base condition any name when assigning it in the service.

Note: Certain conditions can be unavailable in the solution which integrates ERE.

Table 73 The Base Conditions

Condition	Paths
Numeric base conditions, see Section 8.2.1.2 on page 150.	
Short	Class: <code>number.ShortCondition</code> Profile: <code>number.ShortConditionProfile</code>
Integer	Class: <code>number.IntegerCondition</code> Profile: <code>number.IntegerConditionProfile</code>
Long	Class: <code>number.LongCondition</code> Profile: <code>number.LongConditionProfile</code>
Double	Class: <code>number.DoubleCondition</code> Profile: <code>number.DoubleConditionProfile</code>
RatingDecimal	Class: <code>number.RatingDecimalCondition</code> Profile: <code>number.RatingDecimalConditionProfile</code>
UnsignedInt	Class: <code>number.UnsignedIntCondition</code> Profile: <code>number.UnsignedIntConditionProfile</code>
UnsignedInt8	Class: <code>number.UnsignedIntCondition</code> Profile: <code>number.UnsignedInt8ConditionProfile</code>
UnsignedInt16	Class: <code>number.UnsignedIntCondition</code> Profile: <code>number.UnsignedInt16ConditionProfile</code>
UnsignedInt32	Class: <code>number.UnsignedIntCondition</code> Profile: <code>number.UnsignedInt32ConditionProfile</code>



Table 73 The Base Conditions

Condition	Paths
ULong	Class: number.ULongCondition Profile: number.ULongConditionProfile
UInteger	Class: number.UIntegerCondition Profile: number.UIntegerConditionProfile
UShort	Class: number.UShortCondition Profile: number.UShortConditionProfile
In array base conditions, see Section 8.2.1.3 on page 151.	
ValueInArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.ValueInArrayConditionProfile
LongInArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.LongInArrayConditionProfile
IntegerInArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.IntegerInArrayConditionProfile
ShortInArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.ShortInArrayConditionProfile
UnsignedInt32InArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.UnsignedInt32InArrayConditionProfile
UnsignedInt16InArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.UnsignedInt16InArrayConditionProfile
UnsignedInt8InArray	Class: numberseries.ValueInArrayCondition Profile: numberseries.UnsignedInt8InArrayConditionProfile
String base conditions, see Section 8.2.1.4 on page 151.	
String	Class: string.StringCondition Profile: string.StringConditionProfile
OctetString	Class: string.OctetStringCondition Profile: string.OctetStringConditionProfile
StringLength	Class: string.StringLengthCondition Profile: string.StringLengthConditionProfile



Table 73 The Base Conditions

Condition	Paths
NumberString	Class: string.StringCondition Profile: string.NumberStringConditionProfile
BCDString	Class: string.StringCondition Profile: string.BcdStringConditionProfile
RegularExpression	Class: string.RegularExpression Profile: string.RegularExpressionProfile
SubstringAt	Class: string.SubstringAtCondition Profile: string.SubstringAtConditionProfile
Collection conditions, see Section 8.2.1.5 on page 155.	
SetContains	Class: set.SetContainsCondition Profile: set.SetContainsConditionProfile
ValueInCollection	Class: numberseries.ValueInCollectionCondition Profile: numberseries.ValueInCollectionConditionProfile
List base conditions, see Section 8.2.1.6 on page 155.	
StringList	Class: string.StringList Profile: string.StringListProfile
OctetStringList	Class: string.OctetStringList Profile: string.OctetStringListProfile
NumberList	Class: numberseries.NumberSeriesCondition Profile: numberseries.NumberSeriesConditionProfile
NumberListTable	Class: numberseries.NumberSeriesCondition Profile: numberseries.NumberSeriesConditionProfile2
Bit pattern base conditions, see Section 8.2.1.7 on page 158.	
LongBitPattern	Class: bit.BitPatternCondition Profile: bit.LongBitPatternConditionProfile
IntegerBitPattern	Class: bit.BitPatternCondition Profile: bit.IntegerBitPatternConditionProfile
ShortBitPattern	Class: bit.BitPatternCondition Profile: bit.ShortBitPatternConditionProfile
Time and date base conditions, see Section 8.2.1.8 on page 160.	



Table 73 The Base Conditions

Condition	Paths
Date	Class: date.DateCondition Profile: date.DateConditionProfile
Time	Class: time.TimeCondition Profile: time.TimeConditionProfile
Days	Class: date.Days Profile: date.DaysProfile
Other base conditions, see Section 8.2.1.9 on page 163.	
Amount	Class: logic.AmountCondition Profile: logic.AmountConditionProfile
Boolean	Class: logic.BooleanFieldCondition Profile: logic.BooleanFieldConditionProfile
IPAddress	Class: string.IpAddress Profile: string.IpAddressProfile
IPv6Address	Class: string.IpV6AddressCondition Profile: string.IpV6AddressConditionProfile

8.2.1.1 Recurring Concepts in Base Conditions

Certain concepts recur when defining base conditions. To avoid unnecessary repetition some of them are described here.

8.2.1.1.1 Operators Available in Base Conditions

Conditions such as Days, String and the numeric base conditions have operators as parameters. The operators available in these conditions are found in Table 74.

Table 74 Operators Available in Applicable Base Conditions

Possible operators	Description
=	Equal to The condition is true if the configured value is the same as for the value of the associated field.
>	Greater than The condition is true if the value is greater than the parameter selected field.



Table 74 Operators Available in Applicable Base Conditions

Possible operators	Description
<	Less than The condition is true if the value is less than the parameter selected field.
>=	Greater than or equal to The condition is true if the value is greater than or the same as the parameter selected field.
<=	Less than or equal to The condition is true if the value is less that or the same as the parameter selected field.

8.2.1.1.2 Separators in List Base Conditions

The list base conditions compare lists of entries, which may be separated by commas, line breaks or tabulations.

Figure 125 shows an example of entries separated by a comma and a hyphen. The resulting number list is 1, 3, 4, 5, 6.



Figure 125 Example of a List Base Condition

8.2.1.2 Numeric Base Conditions

The numeric base conditions compare the configured value to the value of the associated field, using the specified operator. All the numeric base conditions share the parameters **Operator** and **Value**, see Figure 126. The range of the associated data type limits the range of a numeric base condition.



Figure 126 Example of a Numeric Base Conditions

The operators which may be selected are found in Table 74.

The behavior of the numeric base conditions is the same, except for the **Double** condition. The double values offer an approximation of a floating point number,



but is not exactly equivalent. This may cause unexpected results when calculating numbers with many decimal places.

8.2.1.3 In Array Base Conditions

The in array base conditions check if the entered parameter **Value** exists in the array in the associated field, see Figure 127.

Condition	Value in Array
Value	2

Figure 127 Example of an In Array Base Condition

8.2.1.4 String Base Conditions

The string base conditions share the parameters **Text** and **Comparison**, with some exceptions. The string base conditions compare the configured text to the value of the associated field, using the specified operator, see Figure 128.

Comparison	Begins with
Text	Begins with
Match case	Ends with
	Contains
	Exact match

Figure 128 Comparison Parameter in a String Base Condition

Descriptions of the comparison types are listed in Table 75.

Table 75 Comparisons

Comparisons	Description
Begins with	Checks if the string value of the associated field starts with the configured text.
Contains	Checks if the string value of the associated field contains the configured text.
Ends with	Checks if the string value of the associated field ends with the configured text.
Exact match	Checks if the string value of the associated field is an exact match with the configured text. This condition is case insensitive by default.

As an example of the use of a string base condition, the following may be considered. If the **Comparison** is selected as **Begins with** and the value "aa" is entered in **Text**, the condition is true if the value of the associated is either "Aardvark" or "aardvark".

The **Match case** check box defines if the string condition should be case-sensitive or not. If the check box is selected, the condition in the above example is only evaluated as true if the value of **Selected Field** is "aardvark".



8.2.1.4.1 String

The **String** condition compares the configured text to the value of the associated field, using the specified operator see Figure 129.

Condition	String
Comparison	Begins with
Text	aa
Match case	<input checked="" type="checkbox"/>

Figure 129 Example of the String Condition

Text - the string to be used. The input box accepts any printable characters.

8.2.1.4.2 OctetString

The **OctetString** condition compares the configured Hex or ASCII string text to the value of the associated field, using the specified operator, see Figure 130.

Condition	Octet String
Comparison	Begins with
Text	Hex 68656C6C6F

Figure 130 Example of the OctetString Condition

Text - drop-down menu indicating if the input should be Hexadecimal or ASCII. When changing from ASCII to Hex in the menu, RMA tries to translate the value of the strings to Hex. An error dialog is shown if this translation is not possible.

Next to the drop-down is an input box where the octet string value to be used should be entered. When defined as Hex, only the hexadecimal digits (0-9, A-F) are allowed.

8.2.1.4.3 NumberString

The **NumberString** condition compares the configured numeric string to the value of the associated field, using the specified operator, see Figure 131.



Selected field	String
Condition	Number List
Numbers	

Figure 131 Example of NumberString Condition

Text - the numeric string to be used as input. Only the characters 0–9 can be used.

8.2.1.4.4 BCDString

The **BCDString** condition takes binary coded decimal values in the input field, see Figure 132.

The string is a TBCD (Telephony Binary Coded Decimal). The TBCD string is different from the ordinary BCD string in the way that it allows not only 0–9 but also A–F. The F, however, is only used as a filler and is removed from the editor when saving the configuration.

Condition	BCD String
Comparison	Begins with
Text	

Figure 132 Example of the BCDString Condition

Text - the BCD string to be used as input.

8.2.1.4.5 StringLength

The **StringLength** condition has the parameters **Operator** and **Length**. The condition takes an integer number and uses the operator to compare the number to the number of characters in the string in the assigned field, see Figure 133.

Condition	StringLength
Operator	=
Length	6

Figure 133 Example of the StringLength Condition

Table 74 describes the possible operators in the StringLength condition.

8.2.1.4.6 RegularExpression

The **RegularExpression** condition makes it possible to write a regular expression to be matched with a field holding a String value, see Figure 134.

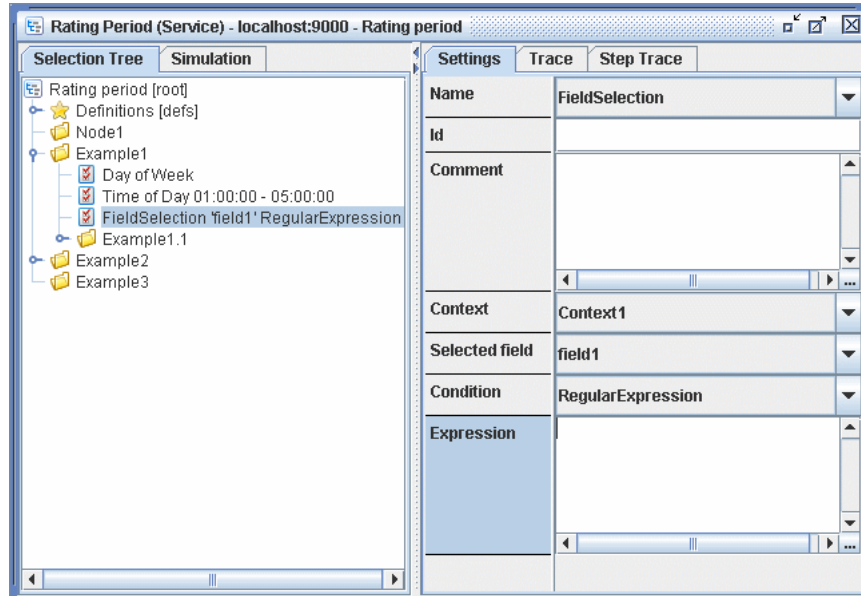


Figure 134 RegularExpression Condition

A regular expression is a string that is used to describe or match a set of strings, according to the syntax rules built into the Java Syntax, described in the Pattern class in the Java API, see <http://docs.oracle.com/javase/8/docs/api/>.

Text: The syntax for a match for a field String. If the syntax is not valid, the text area is red indicating an invalid syntax, see Example 2 and Example 3.

`^[a A].*`

Example 2 Syntax for a Match for a Field String

Will be evaluated as true for a String that begins with a lower or capital case 'a'.

`.*[a A]$`

Example 3 Syntax for a Match for a Field String

Will be evaluated to true for a String that ends with a lower or capital case 'a'.

8.2.1.4.7 SubstringAt

The **SubstringAt** condition has the parameters **Index (0-based)**, **Substring** and **Match Case**. The condition compares the text string configured in substring with the position in the assigned field, defined by the index. The index starts with position 0 as the first character in the string defined in the assigned field.



Index (0-based)	5
Substring	Christmas
Match case	<input checked="" type="checkbox"/>

Figure 135 Example of the SubstringAt Condition

The example in Figure 135 is evaluated as true if the assigned field contains the string Christmas, starting at the sixth letter in the string, for example Cost@Christmas.

8.2.1.5 Collection Conditions

The collection conditions operate on collection fields.

8.2.1.5.1 SetContains

The **SetContains** condition compares the configured value in **Set containing** with the values contained in the associated field. The field must be a set collection. The comparison is made as an exact match, and returns true or false.

8.2.1.5.2 ValueInCollection

The **ValueInCollection** condition compares the value of the associated field, in **Selected field**, with values contained in the associated collection field, in **Collection field**. The comparison can be made as an exact match, or a partial match, and returns true or false. In the field **Operator**, the selectable comparisons are:

- **Starts with any collection entry** - The condition returns true if the value of the associated field starts with a value in the collection.
- **Ends with any collection entry** - The condition returns true if the value of the associated field ends with a value in the collection.
- **Equal to any collection entry** - The condition returns true if the value of the associated field is an exact match of a value in the collection.

8.2.1.6 List Base Conditions

The list base conditions compare the configured lists of values to the value of the associated field, using the specified operator. The lists may be numeric or strings, depending on the base condition used.

8.2.1.6.1 StringList

The **StringList** condition has the parameters **Strings**, **Comparison** and **Match case**. The strings in the list are compared individually to the string in the associated field, using the criteria given in the comparison, as in Figure 136.

Condition	String List
Comparison	Begins with
Strings	<div>Christmas</div> <div>NewYear</div> <div>Midsummer</div>
Match case	<input checked="" type="checkbox"/>

Figure 136 Example of the StringList Condition

Strings - a string containing any ASCII characters.

8.2.1.6.2

OctetStringList

The **OctetStringList** condition has the parameters **Strings**, **Comparison** and **Mode**. The strings in the list are compared individually to the string in the associated field, using the criteria given in the comparison.

Mode - drop-down menu indicating if the input should be Hexadecimal or ASCII. When changing from ASCII to Hex in the menu, RMA tries to translate the value of the strings to Hex. An error dialog is shown if this translation is not possible.

Strings - the Octet values to be used. When defined as Hex, only hexadecimal digits (0-9, A-F) are allowed.

8.2.1.6.3

NumberList

The **NumberList** condition has the parameter **Numbers**. The numbers in the list are compared individually to the value in the associated field, as in Figure 137. The condition accepts both integer and hexadecimal values, using the characters 0-9, A-F, '#' and '*'. The letter characters are not case sensitive. The characters '#' and '*' are equivalent to "C" and "B" respectively.

The condition is based on prefix matching, which means that the conditions match the values that start with the string values in the list. An example of this is that the number list "4 5 68" will match on the values 40, 41, 55, 68, 6800 and so on.

The number "04" in a number list will return a match on the value 041 but not on the value 41. In this case, the associated field must be of the data type string, so that the leading zeros are not discarded when the value is used.

Condition	Number List
Numbers	1, 3-6

Figure 137 An Example of the NumberListCondition



It is possible to define ranges of unsigned integer numbers.

Note: It is strongly advised to keep ranges in the base condition in the same magnitude.

Ranges cannot be defined using hexadecimal values or the characters '#' and '*'.

8.2.1.6.4

NumberListTable

The **NumberListTable** condition has the parameter **Values**. The functionality is the same as for the **NumberList** condition, see Section 8.2.1.6.3 on page 156. **NumberListTable** has a different editor, in which the values are entered in a table variable editor.

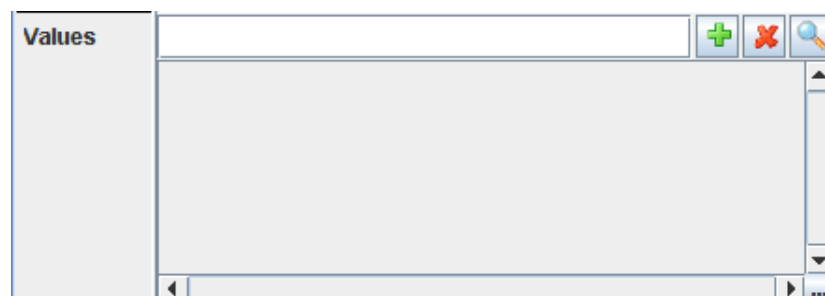



Figure 138 Example of the NumberListTable Condition

Numbers are entered into the list by typing them into the **Values** field and pressing enter or selecting the  button.

If a single number is specified, and a range of number already specified contains that number, a message box prompts if the range should be replaced by the single number. Selecting **Replace** in this box will remove the range from the list, and add the single number instead.

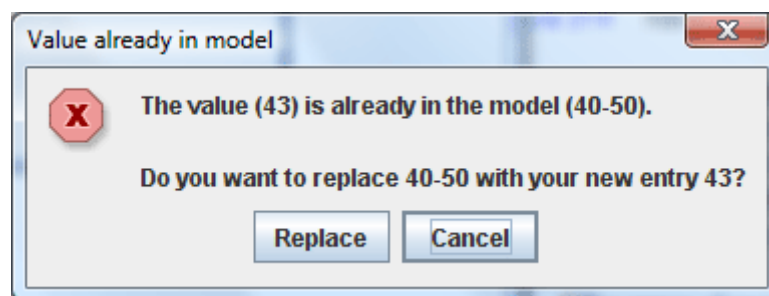


Figure 139 Replace Range Message Box

It is also possible to paste numbers in from the operating systems clipboard, from Notepad or a similar text-based editor. In this way, long and complicated list of numbers and ranges may be entered quickly and easily. The same rules apply for pasted lists as for entries in the **Values** field. If the pasted list contains duplicate entries, an error message will be displayed where it is possible to replace the current entry by choosing **Replace**, or retain the current entry by choosing

Skip. The rest of the entries will continue to be pasted into the number list, and it possible to retain all current entries by choosing **Skip all**.

A **Warning** message box will be displayed at the end of the paste operation, informing on how many duplicate entries were contained in the pasted list. A negative range, for example 50-47, will not be accepted when pasted from the clipboard.

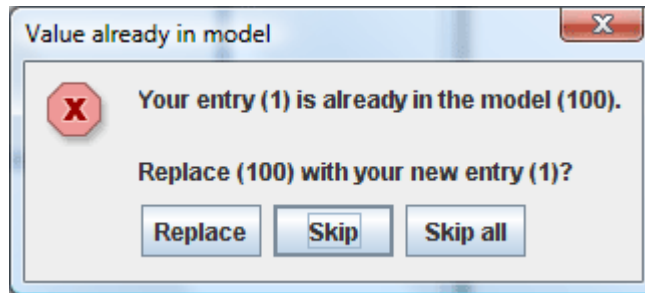


Figure 140 Duplicate Numbers While Pasting from Clipboard Message Box

Numbers in the number list may be deleted by highlighting the number, and pressing delete or selecting the button. Multiple entries may be selected and deleted at the same time.

The list may be searched. Entering the requested number or range, and pressing the **F3** function key or selecting the button will bring up all the entries or ranges containing the request, in a **Search Mode** field.

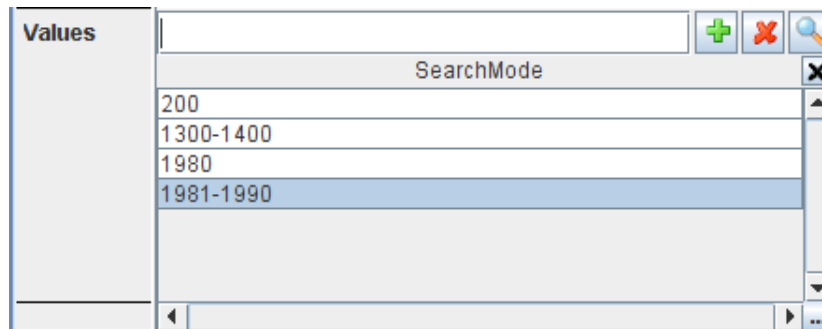


Figure 141 Search Mode Window

The button brings up a new **Values** window, which is identical in form and function to the **Values** field in the condition.

8.2.1.7

Bit Pattern Base Conditions

The **Bit Pattern** conditions have the parameter **Bit Pattern**. The condition performs a bitwise comparison of the configured value against a single or multiple defined bit pattern value.

A bit pattern value is defined in two different ways, they are as follows:



- By a value with corresponding bit mask. The mask makes it possible to define exactly which bits that should be compared with the configured value.
- By a range of values.

The user can add, edit, and delete bit pattern values, see Figure 142. The value is defined by typing in the decimal value or by setting the bits manually.

Bit pattern ID	Active	Mask
1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 142 Edit Bit Pattern Value

By checking the **Define a range** check box, it is possible to define the bit pattern value as a range, see Figure 143.

Bit pattern ID	Active	Mask
1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 143 Define a Range

Note: The bit pattern values are always treated unsigned.



The following are examples of the use of the bit pattern condition.

1. A single Short Bit pattern is defined as a value and a mask. The decimal value is written within the parenthesis:

Bit pattern value: 00000000.11000011 (195)

Bit pattern mask: 00000000.11111111 (255)

2. Input value:

Binary: 00001100.11000011 (3267)

3. The input value is masked with the Bit pattern mask.

Input value: 00001100.11000011 (3267)

Mask: 00000000.11111111 (255)

Masked result: 00000000.11000011 (195)

4. The result is matched against the Bit pattern value:

Masked result: 00000000.11000011 (195)

Bit pattern value: 00000000.11000011 (195)

5. Since the values match the condition returns TRUE.

In another example, a single Short range is defined with a start and an end value.

1. Range values: Start value: 00000000.00001000 (8)

End value: 00000000.00001111 (15)

2. Input value: Binary: 00000000.00011000 (24)

3. The input value is matched against the defined range Start value <= Input value <= End value: 8 <= 24 <= 15

4. Since the input value is greater than the defined end value, the condition returns FALSE.

8.2.1.8

Time and Date Base Conditions

The time and date base conditions compare the configured values to the value of the associated field, using the specified operator. The associated field in this case contains a time or a date.



8.2.1.8.1 Date

The **Date** condition has the parameters **Operator** and **Date**. The condition compares the configured date to the date in the associated field, using the specified operator.

The date may be entered manually, or using the date and time editor, see Section 6.7.13 on page 97.


Condition	Date	▼
Operator	=	▼
Date	2010-10-27	

Figure 144 Example of the Date Condition

Table 74 gives the operators available in the condition.

8.2.1.8.2 Time

The **Time** condition has the parameters **Operator** and **Time**. The condition compares the configured time to the time in the associated field, using the specified operator.

The date may be entered manually, or using the date and time editor, see Section 6.7.13 on page 97.



Settings	Trace	Step Trace	
Name	TimeCondition		▼
Id			
Comment	<div></div> <div> <div>◀</div> <div> </div> <div>▶</div> <div>...</div> </div>		
Operator	=		▼
Time	2008-05-23 13:25:17		 

Figure 145 Time Condition

Table 74 gives the operators available in the condition.

8.2.1.8.3 Days

The **Days** condition compares difference in days between the associated field and the **Comparison Date** field with the number of days in the **Days** field, using an operator.

The specific settings, shown in Figure 146, for the Days Condition are as follows:

- **Comparison date:** A drop-down holding date and time fields, see Table 76.
- **Operation:** A logical operation, see Table 74.
- **Days:** An integer number of days, to be used in the logical expression when the condition is evaluated.

Condition	Days	▼
Comparison date	CurrentTime()	▼
Operation	=	▼
Days	0	

Figure 146 Example of the Days Condition

The calculation of days between the two dates is performed by subtracting the date in the associated fields from the comparison date, see Figure 147. This means that a positive result means the comparison date lies in the future, and a negative result means it lies in the past relative to the date on the name parameter.

The formula for the Days condition is written as: **(Ds-Dc) [operator] Days**

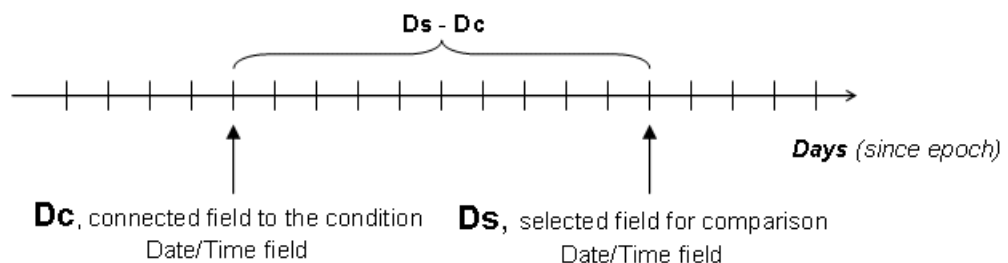


Figure 147 Shows the Subtraction Performed by the Days Condition

An example of how the days condition is evaluated is given here.

The field associated with the condition is set to 31-12-2008, and is called YEAR_END_DATE.

The selected comparison date is the StartTime() (**Ds**) in this example set to 01-12-2008.

To check if the start time is during the year in question, the operation < (less than) is selected.

The number of days is set to 0.

This gives the expression: (StartTime() - YEAR_END_DATE) < 0



$(20081201 - 20081231) < 0$

$-30 \leq 0$, and the condition is evaluated as true

Table 76 Content Description for the Comparison Date Setting

Comparison date	Description
CurrentTime()	Function that returns the current time for the current event. Current time should be used if time unit is used for calculations. If the time unit is not defined, this function returns the same time as the StartTime() function.
StartTime()	Function that returns the time when the current event occurred.
	All associated fields in time or date format are visible in the drop-down list.

8.2.1.9 Other Base Conditions

Some base conditions do not follow the behavior of any of the other the groups of conditions. These are described in this section.

8.2.1.9.1 Amount

The **Amount** condition has the parameters **Operator** and **Amount value**. The condition compares the configured value to the value of the associated field, using the specified operator. Each value consists of a decimal and a currency code.

The configured currency code can be changed in the drop-down menu.

Condition	Amount	▼
Operator	=	▼
Amount value	0	SEK ▼

Figure 148 Example of the Amount Condition

Table 74 gives the operators available in the condition.

8.2.1.9.2 Boolean

The **Boolean** condition makes it possible to evaluate if the associated field is true or false.

There are no parameters in this condition.

8.2.1.9.3 IPAddress

The **IPAddress** condition makes it possible to validate the associated against an IP number and a mask, to determine if the string in the field is an acceptable IPv4 address, see Figure 149.



Selected field	field1	▼
Condition	IP Address	▼
Mask	123.123.123.123/8	

Figure 149 IP Address

The format used for IP addresses is n.n.n.n/x, where:

- n is a part of the IP address in the range 0–255
- x is the mask, in the range 1–32

The IP mask can be described exactly, using a wildcard, or using the Classless Inter Domain Routing (CIDR) format.

Some IPv4 examples of the input accepted in this condition are laid out below.

- **Exact match:** An exact match is to be used when an exact IP address must be compared.

136.0.0.32

- **CIDR:** Using a CIDR mask that the 8 most significant bits for the address given are used for comparison. In this example, an IP address that starts with 123 is considered as a match.

123.0.0.32/8

- **Wildcard Notation:** The IP address can be listed using a wildcard (*), where the wildcard represents any values after its insertion.

136.* is the same as 136.0.0.0/8,
136.225.* is the same as 136.225.0.0/16
and 135.225.10.* is the same as 136.225.10.0/24

8.2.1.9.4

IPv6Address

The **IPv6Address** condition makes it possible to validate an IP address in the associated field against the configured subnet, to determine if the string in the field is a valid IPv4 or IPv6 address within the subnet. Figure 150 shows an example of an IPv6 subnet.

Note: If no CIDR notation is given for a fully qualified address, the **Subnet** field will automatically handle it as an exact match.



Selected field	field1	▼
Condition	IPv6 Address	▼
Version	<input type="radio"/> IPv4 <input checked="" type="radio"/> IPv6	
Subnet	2001:DB8::/32	

Figure 150 IPv6 Subnet

The **IPv6Address** condition is compatible with both IPv4 and IPv6 addresses.

An IPv6 address is written in eight groups, where each group is separated by a colon. Octet series of zeros can be shortened by using two colons. For example, 2001:DB8:85A3:0:0:0:0:1234 may be written as 2001:DB8:85A3::1234.

While an IPv4 address is written in four groups, where each group is separated by a dot.

The format used for IPv6 subnets is n:n:n:n:n:n:n/x, where:

- n is a part of the IP address in hexadecimal digits, in the range 0–FFFF
- x is the CIDR notation, in the range 1–128

The format used for IPv4 Addresses is n.n.n.n/x, where:

- n is a part of the IP address in decimal digits, in the range 0–255
- x is the CIDR notation, in the range 1–32

The IP address can be described with or without a CIDR notation. The **IPv6Address** condition is configured as an exact match if no CIDR notation is given for a fully qualified address.

Some IPv6 examples of the input accepted in this condition are laid out below.

- **Exact match:** An exact match is to be used when an exact IP address must be compared.

2001:DB8:85A3::1234

- **CIDR:** Using a CIDR notation that has the 32 most significant bits in the address used for comparison. In this example, an IP address that starts with 2001:DB8 is considered a match.

2001:DB8:85A3::1234/32

8.2.2

Hierarchical Base Conditions

Hierarchical base conditions are conditions used in a selection tree, where they can be assigned to a specific hierarchical field using the FieldSelection condition, see Section 8.2.3.9 on page 180. A hierarchical base condition can be assigned



to any hierarchical field with matching collection type. Which collection types a hierarchical base condition supports is specified in the service.

The hierarchical base conditions are presented in Table 77, and further described in the sub-chapters. The names in the table are merely suggestions, it is possible to give the instantiated base condition any name when assigning it in the service.

Table 77 Hierarchical Base Conditions

Condition	Paths
CollectionContains	Class: <code>structure.CollectionContainsCondition</code> Profile: <code>structure.CollectionContainsConditionProfile</code>

8.2.2.1

CollectionContains

The **CollectionContains** compares the values of the associated hierarchical field with the configured sub-conditions. The sub-conditions are normal base conditions, that are configured to evaluate any of the data fields that the hierarchical field is composed of.

The configured sub-conditions will be evaluated on every entry in the associated hierarchical collection, until finding a match. Only if all sub-conditions are evaluated to true in the same hierarchical collection entry, the **CollectionContains** condition will be evaluated to true.

FieldSelection is used to select a hierarchical field in **Selected field** together with the CollectionContains condition in **Condition**, to be evaluated according to the criteria in **Subconditions**, see Figure 151.



Name	Field Selection
Id	
Comment	
Selected field	Hierarchical Array
Condition	CollectionContains
Subconditions	Hierarchical Array.ID = 42 Hierarchical Array.Descriptions contains positive
<div> Add Edit Delete </div>	

Figure 151 CollectionContains Condition

Add a sub-condition by pressing the **Add** button and select the hierarchical field in **Name**. The condition used to evaluate the hierarchical field is selected in **Condition**, together with any available comparison method in **Operator**. The value used to evaluate the condition is given, according to the characteristics for the sub-condition, in the input field. For example, the **Integer** condition only allows characters **0-9** as input in **Value**, see Figure 152.



The screenshot shows a standard Windows-style dialog box titled "Add...". It contains four labeled text boxes: "Name" with the text "Hierarchical Array.ID", "Condition" with "Integer", "Operator" with "=", and "Value" with "42". Each text box has a small downward arrow on its right side, indicating a dropdown menu. At the bottom right of the dialog are two buttons: "Ok" and "Cancel".

Figure 152 Add Sub-Condition

8.2.3 Generic Conditions

The generic conditions are described in Table 78.

A generic condition is a condition that does not need the connection to a data field.

Table 78 Generic Conditions

Condition	Paths
CompoundBoolean	Class: <code>logic.CompoundBooleanCondition</code> Profile: <code>logic.CompoundBooleanConditionProfile</code>
DayOfWeek	Class: <code>date.DayOfWeek</code> Profile: <code>date.DayOfWeekProfile</code>
SpecialDates	Class: <code>datelist.DateList</code> Profile: <code>datelist.DateListProfile</code>
TimeOfDay	Class: <code>time.TimeOfDay</code> Profile: <code>time.TimeOfDayProfile</code>
TimeOfCall	Class: <code>time.TimeOfCall</code> Profile: <code>time.TimeOfCallProfile</code>



Table 78 Generic Conditions

Condition	Paths
DoesFieldExist	Class: field.DoesFieldExistCondition Profile: field.DoesFieldExistConditionProfile
CompareField	Class: logic.CompareField Profile: logic.CompareFieldProfile
CompareDateWithMask	Class: logic.CompareDateWithMask Profile: logic.CompareDateWithMaskProfile
FieldSelection	Class: field.FieldSelectionCondition Profile: field.FieldSelectionConditionProfile

8.2.3.1 CompoundBoolean and BooleanOperator

The **CompoundBoolean** condition enables building logical expressions by adding sub conditions. It is also known as the **BooleanOperator** condition in some versions of ERE. The execution considers all the child conditions according to what selection that has been made in the **CompoundBoolean** condition, see Figure 153.

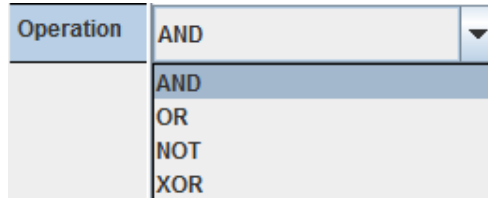


Figure 153 CompoundBoolean Condition

For example, if AND has been selected, all its child conditions must be true for the **CompoundBoolean** condition to be true, see Table 79.

Table 79 CompoundBoolean Condition

Operations	Description
AND	All child conditions must be executed as true.
OR	One of the child conditions must be executed as true for the compound boolean to be true.
NOT	The child condition must be executed as false for the compound boolean to be true.
XOR	Either one, but not both, of the child conditions must be executed as true for the compound boolean to be true. If both child conditions are true, or false, the compound boolean is evaluated to false.

In the figure below, the **CompoundBoolean** condition has four sub conditions, **TimeOfDay** and another **CompoundBoolean** condition containing a **NumberList** and **SpecialDate** condition.

The execution of this condition evaluates this logical expression as in Figure 154.



Figure 154 A Compound Boolean Used in the Tree

8.2.3.2

DayOfWeek

The **DayOfWeek** condition makes it possible to choose the day(s) of a week (Monday-Sunday) for which this condition is true, see Figure 155. Accordingly to this, there is nothing that says that a week must start with Monday and end with Sunday. That is an optional choice.

If a condition is specified for a certain day of the week, there should be no other **DayOfWeek** condition including this day. For example, conditions would overlap if the specified end day in one condition is equal to the start day in another. This condition calculates the time to expiry, and sets the Max time to the time when this condition no longer is valid.

Note: The system gives no warning against specifying nodes with overlapping time-dependent conditions. Overlapping time intervals is confusing, and can possibly have unanticipated charging effects.

From: Drop-down list containing the start day (Monday-Sunday).

To: Drop-down list containing the end day (Monday-Sunday), see Figure 155.

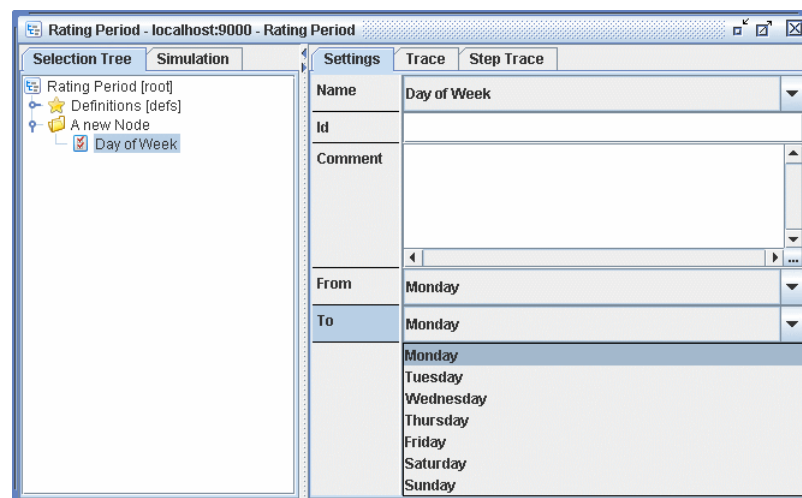


Figure 155 The DayOfWeek Condition



The Figure 156, shows a **DayOfWeek** condition that is executed as true from every Tuesday to Friday in each week.

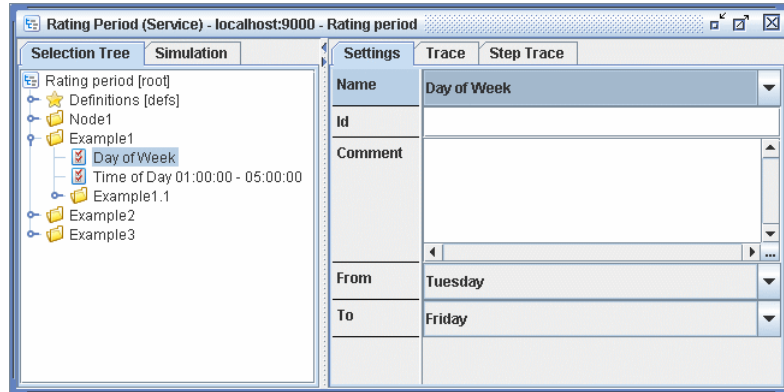


Figure 156 Alternative Choice of Start- and End Day.

The Figure 157, shows a **DayOfWeek** condition that spans from a Sunday in one week, to Thursday the following week. The condition is executed as true for every Sunday to Thursday.

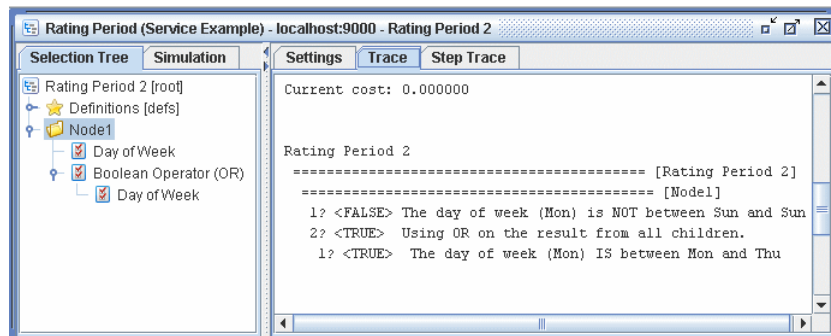


Figure 157 DayOfWeek Condition that Spans from Sunday to Thursday

8.2.3.3

SpecialDates

The **SpecialDates** condition compares the current date against a list of dates. This condition calculates for how long the condition is valid, see Figure 158.

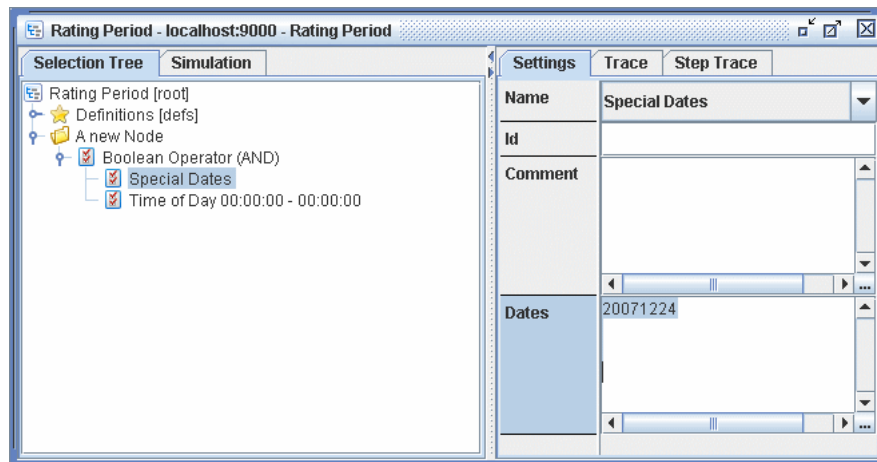


Figure 158 The SpecialDates Condition

The following date format is used; `yyyymmdd`, where:

- `yyyy` equals the year, range 0000–9999.
- `mm` equals the month, range 01–12.
- `dd` equals the day, range 01–31.

The dates can be entered in any of the following formats:

- Single entry: `yyyymmdd`
- Range of dates: `yyyymmdd-yyyymmdd`
- Wildcards can be used for year, month, or day: `yyyymm*`, `*mmdd`, `yyyy*dd`. Wildcards cannot be used in ranges.
- Entries separated by commas or by line breaks.

In the Figure 159, a range is used along with single date entries separated by commas.

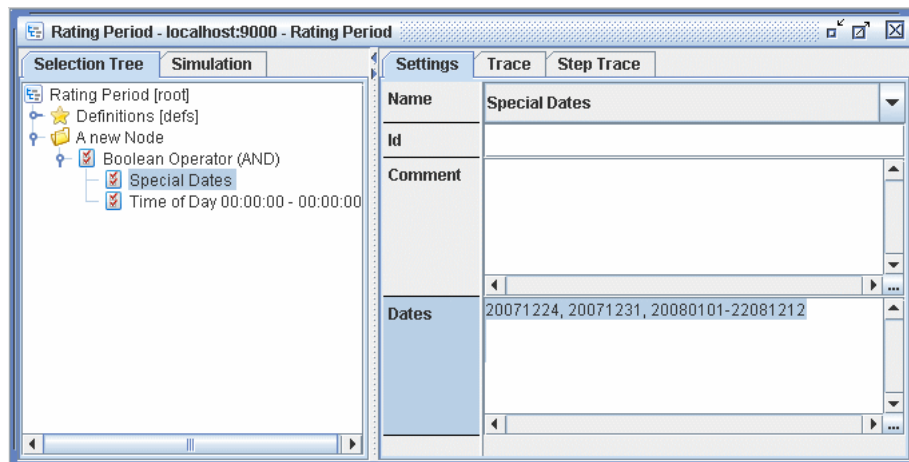


Figure 159 How to Use SpecialDates

In the Figure 160, line breaks are used to separate the different cases.

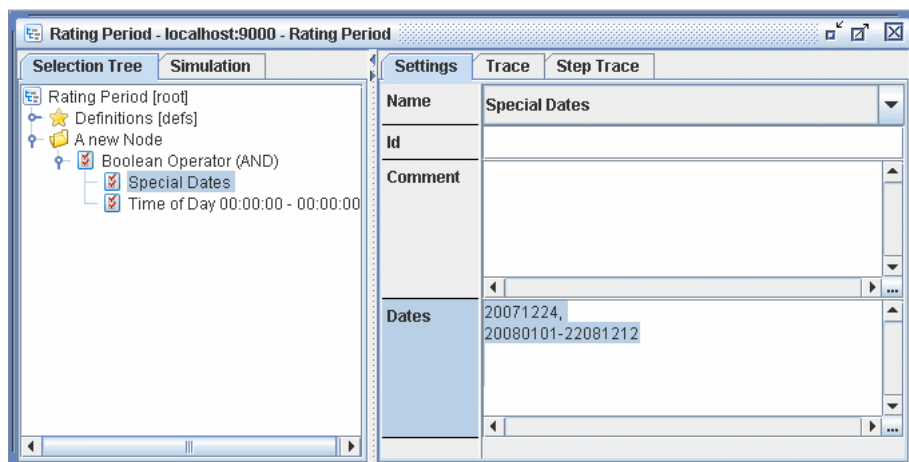


Figure 160 How to Use Line Breaks When Submitting Several Special Dates

8.2.3.4

TimeOfDay

The **TimeOfDay** condition, see Figure 161, makes it possible to choose the time of a day (00:00:00 – 23:59:59) for which this condition is true. An example is presented as follows.

Example 4 To set a charge from midnight until six o'clock in the morning, set the start time to 00 00 00 and the end time to 05 59 59, and connect a modifier to this condition. Then it is possible to create a TimeOfDay condition with the start time 06 00 00, and so on. This condition can have the same start time and end time, which allows a single second to be rated separately.

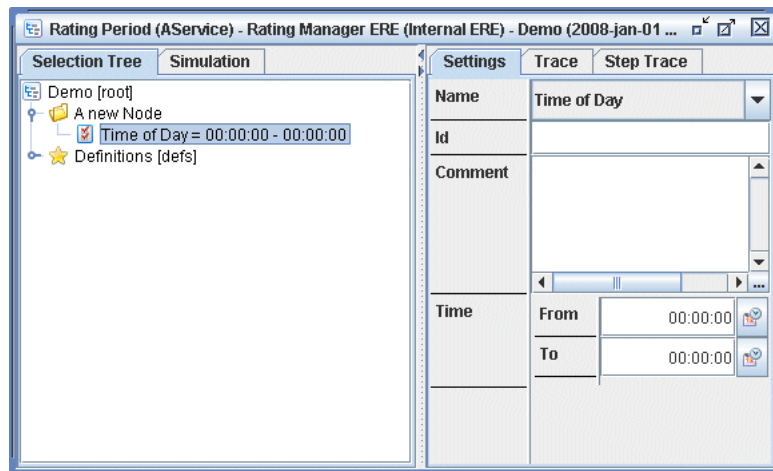


Figure 161 The TimeOfDay Condition

Note: The system gives no warning against specifying nodes with overlapping time-dependent conditions. Overlapping time intervals is confusing and can possibly have unanticipated charging effects.

From: The time when the condition starts to be valid.

To: The time when the condition is not valid.

The time format used is hh:mm:ss

- hh – the hour of the day, range: 00-23
- mm – the minute of the hour, range: 00-59
- ss – the second of the minute, range: 00-59

8.2.3.5

TimeOfCall

The **TimeOfCall** condition, see Figure 162, makes it possible to specify points of time in a call when the condition is to be started and ended. The condition is used to take new actions within the call time. This condition can have the same start time and end time, which allows a single second to be rated separately.

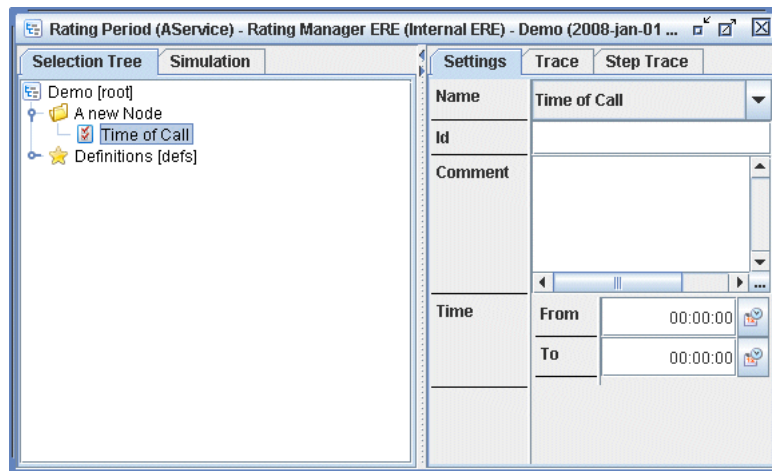


Figure 162 TimeOfCall Condition

From: The time when the condition starts to be valid

To: The time when the condition ends being valid

The time format used is hh:mm:ss

- hh – the hour of the day, range: 00-23
- mm – the minute of the hour, range: 00-59
- ss – the second of the minute, range: 00-59

For example, to set a special charge during the first two minutes of the call, set the start time to 00 00 00 and the end time to 00 01 59.

8.2.3.6

DoesFieldExist

The **DoesFieldExist** condition checks if data for an associated field exists. The fields available are shown in the **Field** parameter drop-down menu.

Figure 163 shows the condition and an example of an associated drop-down list.

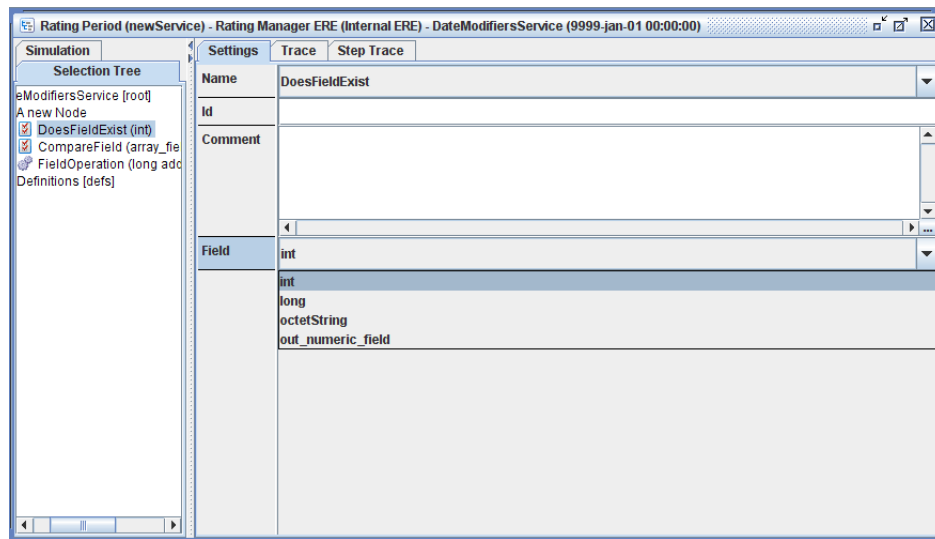


Figure 163 DoesFieldExist Condition

This condition is used to test if the data in a field or an entry in a map field is available for subsequent elements in a selection tree. Hierarchical fields are supported.

If the associated field is a map, a **Check Map Value** check box is visible. Selecting this check box allows a specific value in a map to be chosen as the field to be tested. If the check box is not selected, then the entire map is tested. By using constraints when setting up the service connected to the condition, it is possible to change the behavior of the check map value function. The solution that integrates ERE may already implement these changes. The following changes are possible:

- Hide the check box. This forces the condition to evaluate if the entire map exists, instead of a single value.
- Force the check box to be selected. This forces the condition to evaluate if a single value in the map exists, rather than if the entire map exists.

For more information on how to use constraints, see Section 5.5 on page 60.

8.2.3.7

CompareDateWithMask

The condition **CompareDateWithMask** makes it possible to compare the value of two fields containing dates and validate their values relative to one another and an operator. The comparison uses the day, month or year of the fields or a combination of all three. The condition is shown in Figure 164.



Name	CompareDateWithMask
Id	
Comment	
Field 1	EreCommonOutDate
Operator	=
Compare to field	<input type="checkbox"/>
Value	2011-11-17
Compare using	<input checked="" type="checkbox"/> Year <input checked="" type="checkbox"/> Month <input checked="" type="checkbox"/> Day

Figure 164 CompareDateWithMask Condition

By selecting a field in the first drop-down, **Field 1**, all fields of with the data type Date are available for comparison with are visible in **Field 2**.

The **Operator** drop-down shows the different available options, see Table 80.

Table 80 Operator Options

Operator	Description
=	Equals
>	Greater than
<	Less than
>=	Greater than or equals
<=	Less than or equals

By selecting the **Compare to field** check box, Field 1 is compared to another date field in Field 2. If **Compare to field** has not been selected, a date may be entered into the Value parameter. This Value can be entered manually, or using the Date editor.

The mask is made up of three **Compare using** check boxes, **Year**, **Month**, and **Day**.

Checking these define which part of the date fields to compare. Choosing the **Year** check box compares only the year part of the dates contained in Field 1 and Field 2. A combination of check boxes can also be chosen, for example the day and month of the date fields can be compared.

8.2.3.8 CompareField

The condition **CompareField** makes it possible to compare the value of two data fields of the same data type and validate their values, see Figure 165.

By selecting a field in the first drop-down, **Field 1**, all fields of the same data type available for comparison with are visible in **Field 2**.

If the data field is an array, it is possible to define a specific indexed value in the array to be compared to a similarly specified index value in another field. This is supported for numerical entries of the indexed value. The indexed value may be defined manually, or by using the check boxes in the condition dialog.

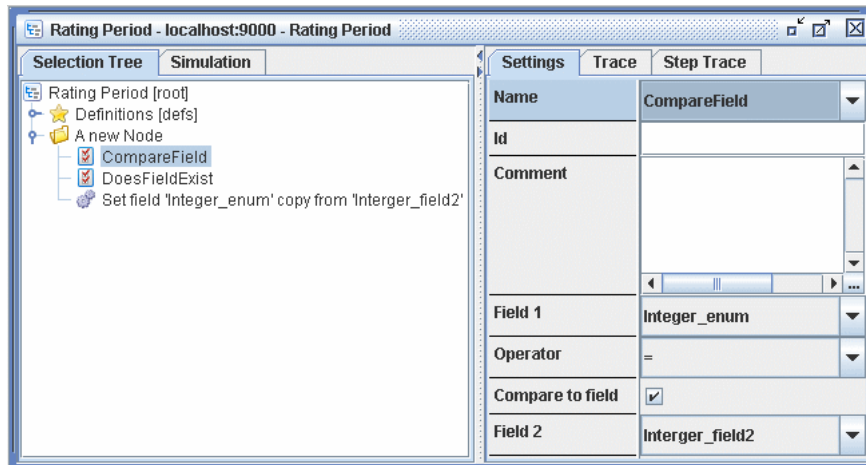


Figure 165 Comparing Field Conditions

Field 1 contains the fields defined in the service, configured as a value representing a single value field or a value list field.

The **Operator** drop-down shows different options depending on the data type, see Figure 166.

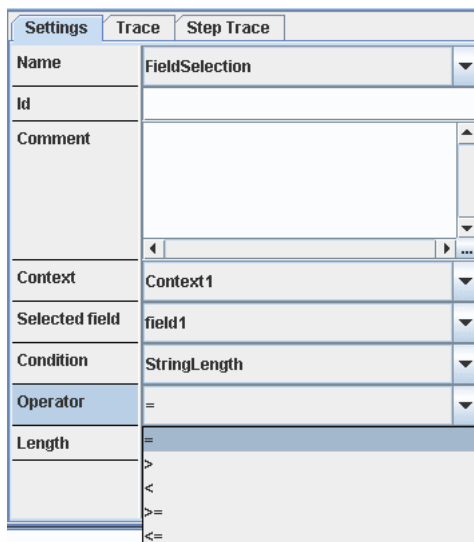


Figure 166 Different Options in the Operator Drop-down

The different operators available are found in Table 81:



Table 81 Operator Options

Boolean	String	Numerical
= (equals)	Begins_with	= (equals)
	Ends_with	> (greater than)
	Contains	< (less than)
	Exact_match	>= (greater than or equals)
	Exact_match	<= (less than or equals)

By selecting the **Compare to field** checkbox, the user can choose to compare **Field 1** to another defined field in **Field 2**. If **Compare to field** has not been selected, there is a parameter available where the desired value to compare with is entered, see Figure 167. The value entered is validated to be in range, and allowed input for the data type of the field in **Field 1**.

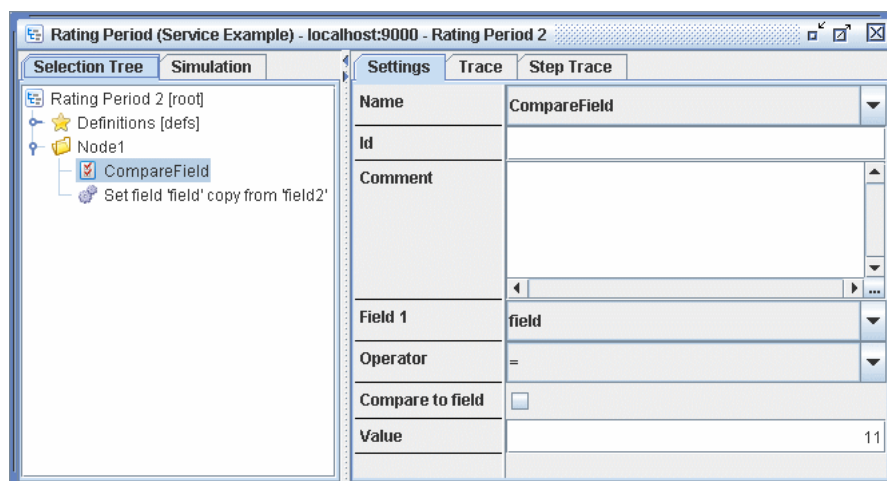


Figure 167 Input Value

In the case where the Field 1 is a value list field, an additional drop-down, an **Index for field 1** menu, will appear. The drop-down will show the defined indexes of the value list. Field 2 can also contain either a single value field or a value list field. In this case an **Index for field 2** menu will show the available indexes. Figure 168 shows an example of this.



Field 1	array_field
Index for field 1	<input checked="" type="checkbox"/> Use field
	out_numeric_field
Operator	=
Compare to field	<input checked="" type="checkbox"/>
Field 2	array_field
Index for field 2	<input checked="" type="checkbox"/> Use field
	int

Figure 168 Example of CompareField Condition When Field 1 and Field 2 Are Value List Fields.

8.2.3.9 FieldSelection

The **FieldSelection** condition makes it possible to use a base condition as the data type of the selected field. Hierarchical fields are supported.

When a field, which may be an SDF or a TDF, is selected, the possible conditions for a field of that data type appear in the **Condition** drop-down. By selecting a condition, the possible input parameters for that condition appear and have the same parameters as the condition itself.

The behavior of **FieldSelection** can be constrained for all fields individually. The field is enabled in **FieldSelection** by configuring the data field as **Show in plugins** in the service editor, see Section 5.3.6 on page 40, or by defining a constraint, Section 5.5 on page 60.

When a field is selected in the **FieldSelection** condition, the possible conditions for the data type of the field are selectable in the **Condition** drop-down, see Figure 169.

If the data type for a field is an array, individual entries in the array may be specified.

Selected field	Number 1
Condition	Integer
Operator	Integer
Value	Long
	IntegerBitPattern
	Double
	Number List

Figure 169 Condition Drop-down

The **Integer** base condition is selected, and the parameters **Operator** and **Value**, that belongs to this condition are visible, see Figure 170. By selecting the **Operator** = and **Value** 10, the **FieldSelection** condition evaluates that the field **Number1** is equal to value 10 in execution.



Selected field	Number 1	▼
Condition	Integer	▼
Operator	=	▼
Value	10	

Figure 170 Integer Base Condition

Selected Field:

Drop-down showing all SDFs and TDFs, that exist.

Index:

If the field selected is an array or a map, an index parameter is visible in the GUI. By entering an index value, the value in that position of the array or map will be compared to the value entered in **Value**.

Selected field	array_field	▼
Index	<input type="checkbox"/> Use field 0	
Condition	Long	▼
Operator	=	▼
Value	0	

Figure 171 Array Condition with Index Field

Condition:

Drop-down showing all conditions possible to use for the data type of the selected field.

Other input depends on what condition that is selected. See possible conditions in Table 82.

Table 82 Conditions in the Drop-down

Conditions
Short Condition
Integer Condition
Long Condition
Double Condition
UnsignedInt Condition
UnsignedInt8 Condition
UnsignedInt16 Condition
UnsignedInt32 Condition
ULong Condition



Table 82 Conditions in the Drop-down

Conditions
UInteger Condition
UShort Condition
ValueInArray Condition
LongInArray Condition
IntegerInArray Condition
ShortInArray Condition
UnsignedInt32InArray Condition
UnsignedInt16InArray Condition
UnsignedInt8InArray Condition
String Condition
OctetString Condition
StringLength Condition
NumberString Condition
BCDString Condition
StringList Condition
OctetStringList Condition
NumberList Condition
Boolean Condition
IPAddress Condition
IPv6Address Condition
LongBitPattern Condition
IntegerBitPattern Condition
ShortBitPattern Condition
RatingDecimal Condition
Amount Condition
Date Condition
Time Condition
Days Condition
RegularExpression Condition



8.3 Modifiers

The modifiers are presented in Table 83 and further described in the sub-chapters below. The qualifications described below all have `com.ericsson.ere.selectiontree.modifiers` as the first part of the qualification. For example Rate modifier is fully qualified as `com.ericsson.ere.selectiontree.modifiers.Rate`, with the class `com.ericsson.ere.selectiontree.modifiers.RateProfile` as the profile.

Table 83 Standard Modifiers

Modifier	Paths
Rate	Class: Rate Profile: RateProfile
RateField	Class: RateField Profile: RateFieldProfile
RateInterval	Class: RateInterval Profile: RateIntervalProfile
RateFieldInterval	Class: RateFieldInterval Profile: RateFieldIntervalProfile
Fee	Class: Fee Profile: FeeProfile
MultiplyFee	Class: MultiplyFee Profile: MultiplyFeeProfile
OneHitFee	Class: OneHitFee Profile: FeeProfile
OneHitMultiplyFee	Class: OneHitMultiplyFee Profile: multiply.MultiplyFeeProfile
SetField	Class: SetFieldModifier Profile: SetFieldModifierProfile
FieldOperation	Class: FieldOperation or Class: FieldOperation2 Profile: FieldOperationProfile
ExitTree	Class: ExitTreeModifier Profile: ExitTreeModifierProfile

Table 83 Standard Modifiers

Modifier	Paths
GoTo	Class: jump.GoToModifier Profile: jump.GoToModifierProfile
ClearValue	Class: ClearValue Profile: ClearValueProfile
MultiFieldOperator	Class: MultiFieldOperation Profile: MultiFieldOperationProfile
DateDifference	Class: MultiFieldOperation Profile: DateDifferenceProfile
DateOffset	Class: MultiFieldOperation Profile: DateOffsetProfile
BreakPoint	Class: BreakPointModifier Profile: BreakPointModifierProfile

8.3.1 Rate

The **Rate** modifier holds the price to be charged per minute. The rate is also accompanied with a charging interval, which is the time in seconds on which charging is based, see Figure 172. An event is charged for each initiated charging interval. This modifier can only be used together with time-based services.

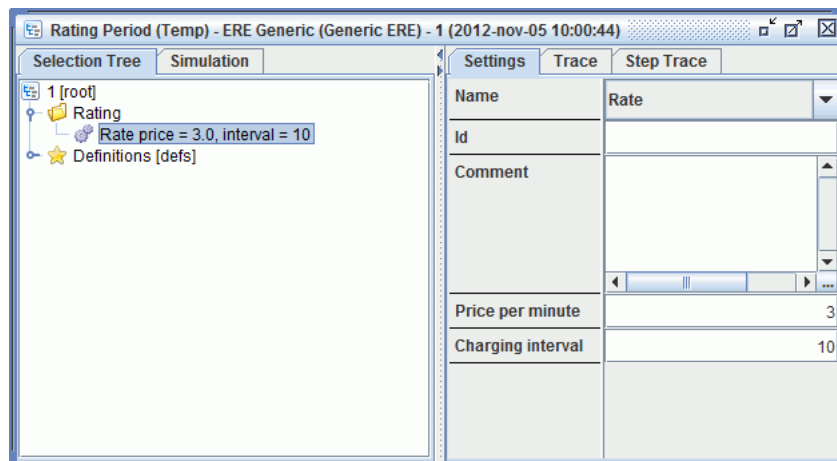


Figure 172 Rate Modifier

Price per minute – Monetary units per minute. A decimal value may be used.



Charging interval – Time interval for charging in seconds. An integer value may be used. The value must be 1 or greater.

If the **Charging interval** is set to 10 and the **Price per minute** is set to 3. This means that the event is charged with 0.5 monetary units, $(3 / 60 * 10)$, for each initiated charging interval of 10 seconds.

Note:

- For the Rate modifier to work, the **DataSet Factory** and **Simulation Factory** must be set to in the Service Element of the connected Service. This gives the monetary and duration awareness in seconds that is needed for performing rating in the selection tree and simulation.
 - **DataSet Factory** - `com.ericsson.ere.dataset.RatingDataSetFactory`
 - **Simulation Factory** - `com.ericsson.ere.gui.simulator.GroupedRatingDataSetSimulationFactory`
- After rating of all units, the selection tree execution stops once the **Rate** modifier is hit.

8.3.2

RateField

Similar to the **Rate** modifier. An event is charged for each initiated charging interval. The **RateField** modifier supports decimal numbering in **Price per minute** and **Charging interval** and fields may be used to provide the values.

An explanation of used parameters:

Price per minute - Monetary units per minute. A decimal value or a numeric field may be used.

Charging interval – Time interval for charging in seconds. A decimal value or a numeric field may be used. The value must be greater than 0.001 (1 ms). Maximum of three decimals may be used. If a field is used to provide the value, it is truncated at three decimals (milliseconds).

If the **Charging interval** is set to 0.5 and the **Price per minute** is set to 3.99. This means that the event is charged with 0.033 monetary units, $(3.99 / 60 * 0.5)$, for each initiated charging interval of 0.5 seconds.

Note:

- For the **RateField** modifier to work, the **DataSet Factory** and **Simulation Factory** must be set to in the Service Element of the connected Service. This gives the monetary and duration awareness in milliseconds that is needed for performing rating in the selection tree and simulation.
 - **DataSet Factory** - `com.ericsson.ere.dataset.MillisecondAllocatableRatingDataSetFactory`
 - **Simulation Factory** - `com.ericsson.ere.gui.simulator.GroupedMillisecondAllocatableRatingDataSetSimulationFactory`
- After rating of all units, the selection tree execution stops once the **RateField** modifier is hit.

8.3.3

RateInterval

The **RateInterval** modifier holds the price to be charged per price interval. The rate is also accompanied with a charging interval which is the time in seconds on which charging is based, see Figure 173. An event is charged for each initiated charging interval.

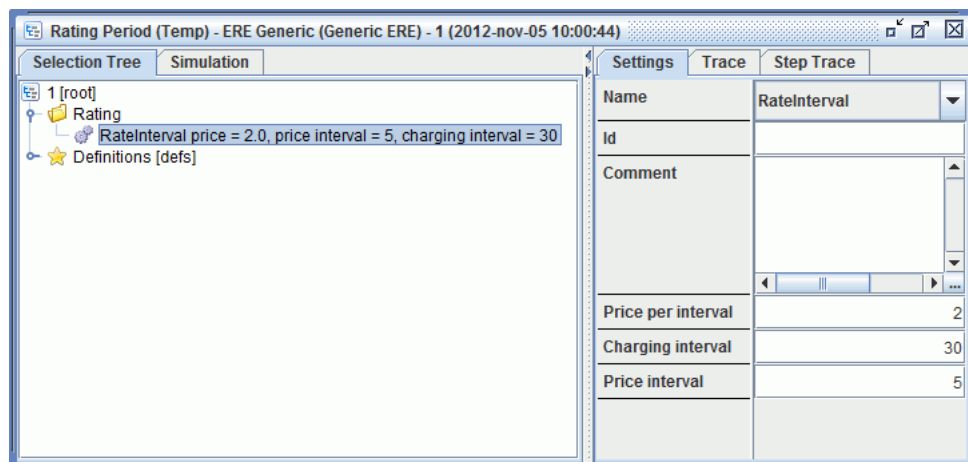


Figure 173 RateInterval Modifier

Price per interval – Monetary units per price interval. A decimal value may be used.

Price interval – Length of the price interval in seconds. An integer value may be used. The value must be 1 or greater.

Charging interval – Time interval for charging in seconds. An integer value may be used. The value must be 1 or greater.

The value provided in **Charging interval** states how often the service is charged. The values provided in **Price per interval** and **Price interval** states the charged



cost. If the **Charging interval** is set to 30, **Price per interval** is set to 2 and **Price interval** is set to 5. This means that the event is charged with 12 monetary units, $(2 / 5 * 30)$, for each initiated charging interval of 30 seconds.

Note:

- For the **RateInterval** modifier to work, the **DataSet Factory** and **Simulation Factory** must be set to in the Service Element of the connected Service. This gives the monetary and duration awareness in seconds that is needed for performing rating in the selection tree and simulation.
 - **DataSet Factory** - `com.ericsson.ere.dataset.RatingDataSetFactory`
 - **Simulation Factory** - `com.ericsson.ere.gui.simulator.GroupedRatingDataSetSimulationFactory`
- After rating of all units, the selection tree execution stops once the **RateInterval** modifier is hit.

8.3.4 RateFieldInterval

Similar to the **RateInterval** modifier. An event is charged for each initiated charging interval. The **RateFieldInterval** modifier supports decimal numbering in **Price per interval**, **Charging Interval**, and **Price interval**. Numeric fields may be used to provide the values.

An explanation of used parameters:

Price per interval – Monetary units per price interval. A decimal value or a numeric field may be used.

Price interval – Length of the price interval in seconds. A decimal value or a numeric field may be used. The value must be greater than 0.001 (1 ms). Maximum of three decimals may be used. If a field is used to provide the value, it is truncated at three decimals (milliseconds).

Charging interval – Time interval for charging in seconds. A decimal value or a numeric field may be used. The value must be greater than 0.001 (1 ms). Maximum of three decimals may be used. If a field is used to provide the value, it is truncated at three decimals (milliseconds).

The value provided in **Charging interval** states how often the service is charged. The values provided in **Price per interval** and **Price interval** states the charged cost. If the **Charging interval** is set to 10, **Price per interval** is set to 1.5 and **Price interval** is set to 0.5. This means that the event is charged with 30 monetary units, $(1.5 / 0.5 * 10)$, for each initiated charging interval of 10 seconds.

Note:

- For the **RateFieldInterval** modifier to work, the **DataSet Factory** and **Simulation Factory** must be set to in the Service Element of the connected Service. This gives the monetary and duration awareness in milliseconds that is needed for performing rating in the selection tree and simulation.
 - **DataSet Factory** - `com.ericsson.ere.dataset.MillisecondAllocatableRatingDataSetFactory`
 - **Simulation Factory** - `com.ericsson.ere.gui.simulator.GroupedMillisecondAllocatableRatingDataSetSimulationFactory`
- After rating of all units, the selection tree execution stops once the **RateFieldInterval** modifier is hit.

8.3.5 Fee Modifiers

This section describes the two fee modifiers.

8.3.5.1 Fee

The **Fee** modifier adds a fixed amount to the cost for the service, see Figure 174.

Note: For time-based services, valid rates must exist for all parts of the rated service. If a fee is used, it must be accompanied by a rate which covers the whole deducted period.

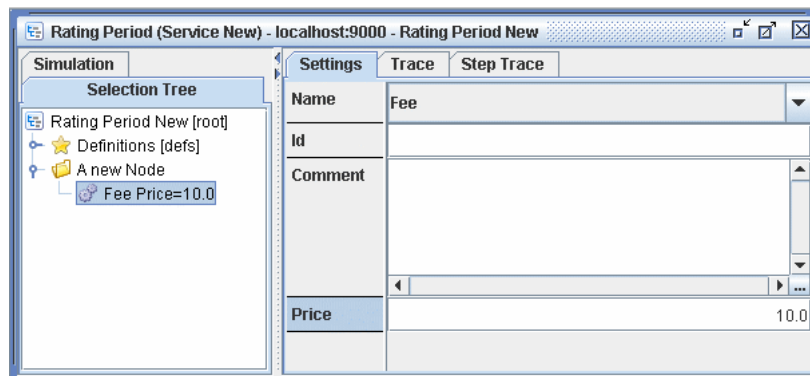


Figure 174 Fee Modifier

Price – the amount of the fee. Range: decimal values (monetary units).

**Note:**

- For the **Fee** modifier to work, the Data set factory **RatingDataSetFactory** needs to be entered in the **DataSetFactory** field in the Service Element of the connected Service. This gives the monetary and duration awareness to the Data Set that is needed for performing rating in the Selection Tree.
- If the fee calculated is more than available balance, the selection tree execution stops once the **Fee** modifier is hit.

8.3.5.2 OneHitFee

A one hit fee adds the fee to the total sum, and consumes all remaining time for the rating request. That means, that a one hit fee can only be applied once in a rating request. A one hit fee must be the last hit in the logical branch of the selection tree.

Price – the amount of the one hit fee. Range: decimal values (monetary units).

8.3.6 Multiply Fee Modifiers

This section describes the two multiply fee modifiers.

8.3.6.1 MultiplyFee

The **MultiplyFee** modifier can be used to multiply the fee with a numeric field value.

As an example of calculating a fee based on the size of a message, a 100 bytes message at a price per byte is 0.2 monetary units results in a fee of 20.

Price – range: decimal values (monetary units).

Field – drop-down menu containing the SDF with which multiplication takes place.

Note: If the fee calculated is more than available balance, the selection tree execution stops once the **Multiplyfee** modifier is hit.

8.3.6.2 OneHitMultiplyFee

A one hit multiply fee adds the fee to the total sum, and consumes all remaining time for the rating request. That means that a one hit multiply fee can only be applied once in a rating request. A one hit multiply fee must be the last hit in the logical branch of the selection tree.

Price – range: decimal values (monetary units).

Field – drop-down menu containing the fields of the service with which multiplication can take place.

8.3.7

SetField

The **SetField** modifier enables setting a field using a value, or copying the value to the selected field from an SDF or TDF. Hierarchical fields are supported. All fields, except fields with parameter type IN, can be set with this modifier, see Figure 175.

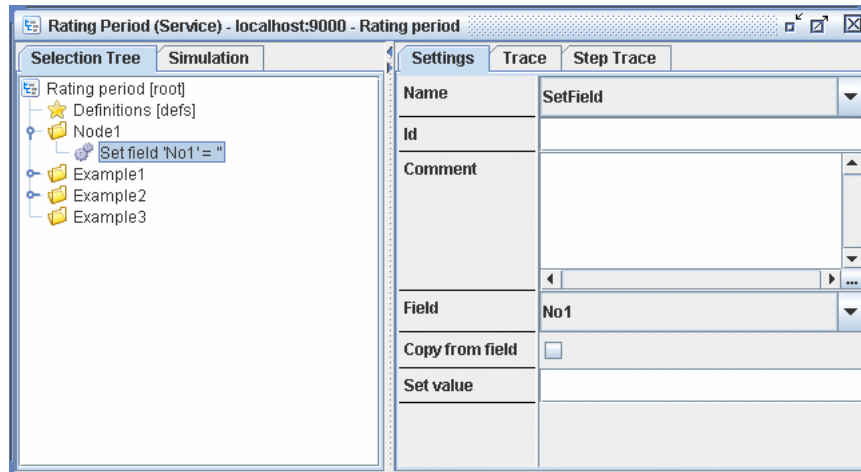


Figure 175 Set Field Modifier

If no fields are defined for the structure, the **SetField** modifier shows **No available fields found**.

A listed field can be given a value in the **Set value** field. If a range is set in the Simulation Data, in creation of the field, the input validates the entered value to be within the specified range.

If **Enumeration** is chosen in the creation of the field, all the names stated there are shown in a drop-down at the field Set value, see Figure 177. The desired value is selected in the drop-down and no other values than these are possible to set. Each value stated in the enumeration is represented by the name, the value connected to it cannot be seen.

If **Copy from field** is selected, it is possible to copy the value from one field to another. The Set value field is replaced with a drop-down, where all possible fields to copy the value from are shown, see Figure 176. These are fields of compatible data types. It is not possible to copy the value from a field that is not of the same data type.

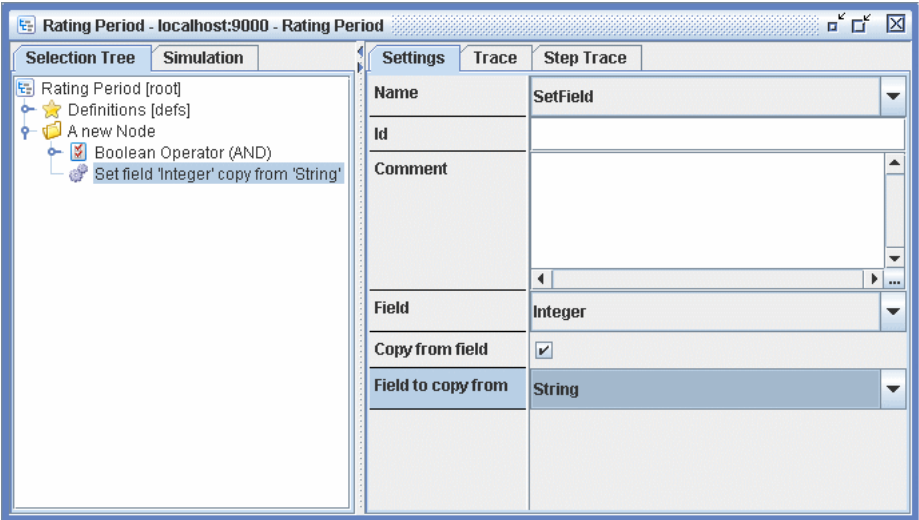


Figure 176 Copy a Field

Field	No2
Copy from field	<input type="checkbox"/>
Set value	A
	A
	B
	C

Figure 177 Enumeration Drop-Down

8.3.8 FieldOperation

The **FieldOperation** modifier makes it possible to perform an operation on a field taking an SDF or a TDF as the **Field to affect**.

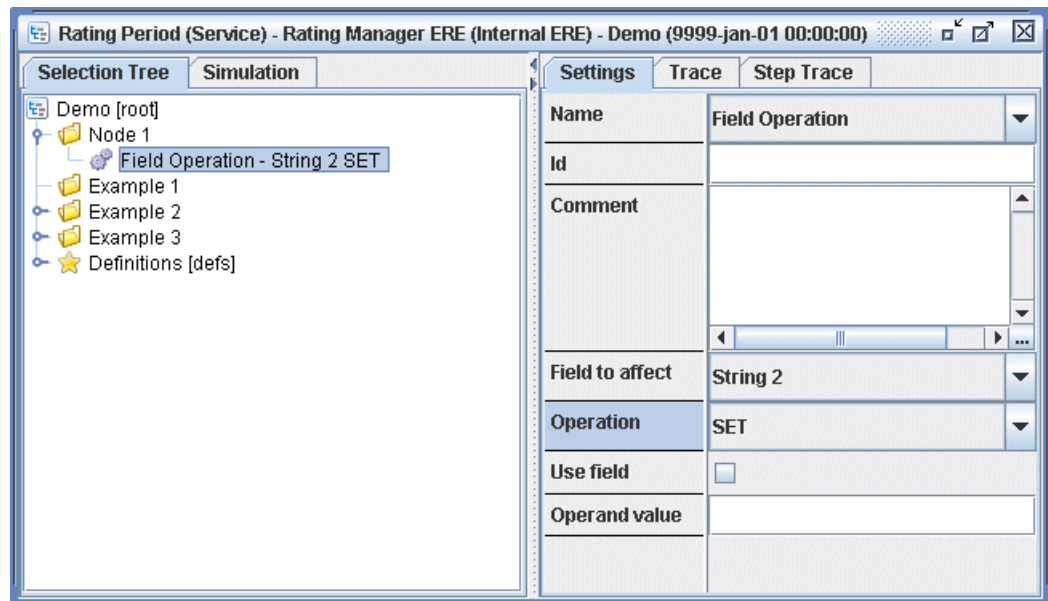


Figure 178 FieldOperation Modifier

The **Operand** may be an indexed field from a value list. In this case, it will be possible to set an **Operand field index**. It is then possible to choose either a defined field or to enter a value for the index, see Figure 179.

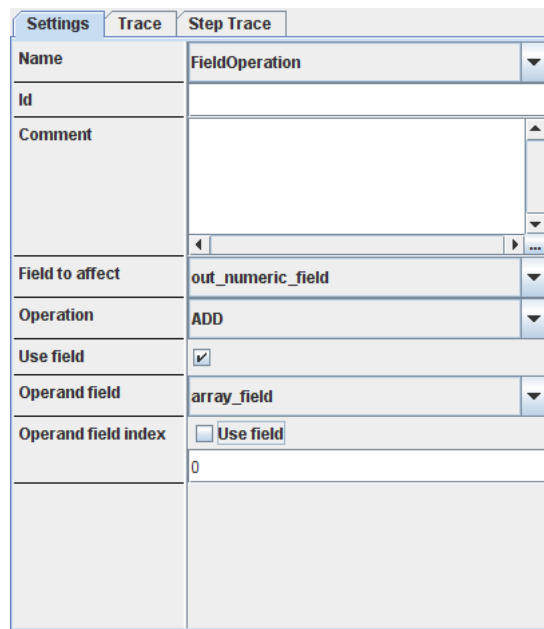


Figure 179 FieldOperation with a Field from a Value List.

If no SDFs or TDFs have been defined, no fields are found and the **FieldOperation** modifier shows **No available fields found**.

There are different operations available depending on the data type of the chosen field. Table 84 explains the available operations.



Table 84 Available Field Operations

Data type	Operations
Numeric	<p>Set – sets the chosen field with a numeric value.</p> <p>Add – adds a numeric value to the chosen field.</p> <p>Subtract – subtracts a numeric value from the chosen field.</p> <p>Multiply – multiplies the chosen field with a numeric value.</p> <p>Divide – divides the chosen field with a numeric value.</p> <p>Note: If the calculation of the operation is less than the minimum of the field or greater than maximum of the field the resulting value will be wrapped around. For example, if a field has the minimum value of 1 and maximum value of 10 and the operation is 5 ADD 6 the result will be 1. This is the default behavior in ERE but in different integrated solutions this behavior may differ.</p>
String	<p>Set – sets the chosen field with a String.</p> <p>Append – adds a String to the end of the chosen fields string value.</p> <p>Prepend – adds a String to the beginning of to the chosen fields string value.</p>
Boolean	Set – sets the chosen field with a Boolean value.
Date	<p>Set – sets the chosen field with a Date value.</p> <p>Add – adds a value to the chosen field. Available units are: Year, month and day.</p> <p>Subtract – subtracts a value from the chosen field. Available units are the same as for Add.</p>
Time	<p>Set - Sets the chosen field with a Time value.</p> <p>Add - Adds a value to the chosen field. Available units are: Year, month, day, hour, minute and second.</p> <p>Subtract - Subtracts a value from the chosen field. Available units are the same as for Add.</p>
Amount	<p>Set - Sets the chosen field with a value.</p> <p>Add - Adds a value to the chosen field.</p> <p>Subtract - Subtracts a value from the chosen field.</p>

Note: Operations do not exist for all data types. Field operation does not support all data types in ERE.

If **Use field** is not selected it is possible to define a fixed value as an operand for the operation, see Figure 178.

If **Use field** is selected it is possible to chose any field of a comparable type as operand for the operation, see Figure 180.

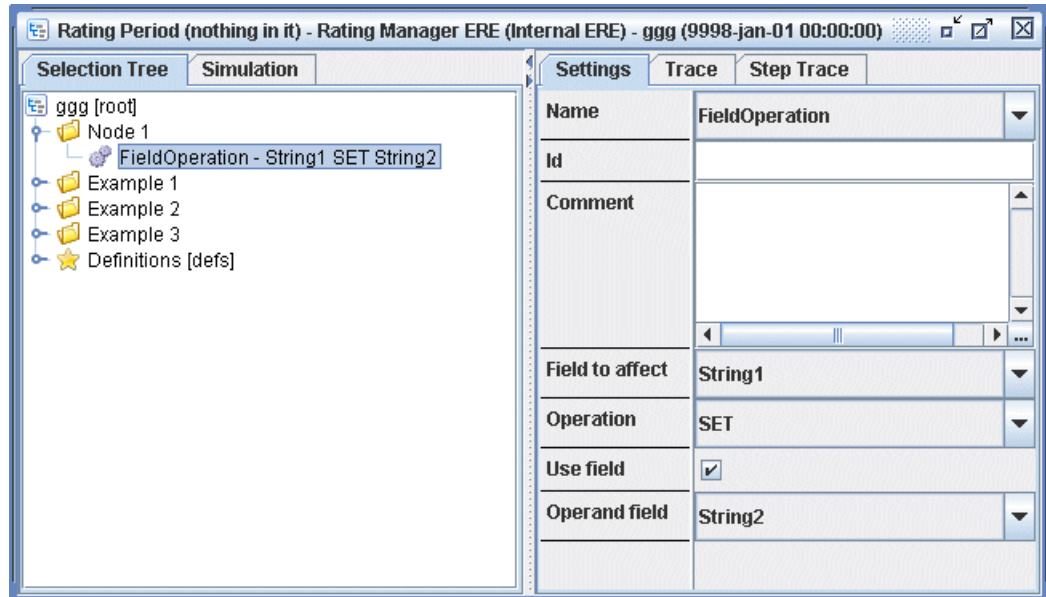


Figure 180 FieldOperation - Use Field

The field operation modifier may be used to modify the variables **Beginning of Time** and **End of Time**, see Section 5.3.12 on page 49.

8.3.9 ExitTree

The **ExitTree** modifier stops the selection tree execution.

8.3.10 GoTo

The **GoTo** modifier enables a jump from the selection tree executing for the moment, to another selection tree in a rating period, with a start date valid for the moment.

Preferences for a jump to be performed are as follows:

- The Selection Tree to jump from must have the same Service as the Selection Tree to jump to.
- The Selection Tree to jump to must have a start date valid for the moment. This also means that jumping to a Rating Period in the same Rating Plan is impossible, since only one Rating Period in a Rating Plan can have a start date valid for the moment.
- It is not possible to jump to the initial point again. The reason for this is to avoid circular execution. This is only checked in runtime (execution of the tree) and results in a Tree Execution Exception. If a chain of jumps is made,



it is not allowed to jump to a tree that has earlier been in the jump series, of the same reason.

The selections in the drop-down show the possible targets in the form Rating Period name(Service)Rating Plan, see Figure 181.

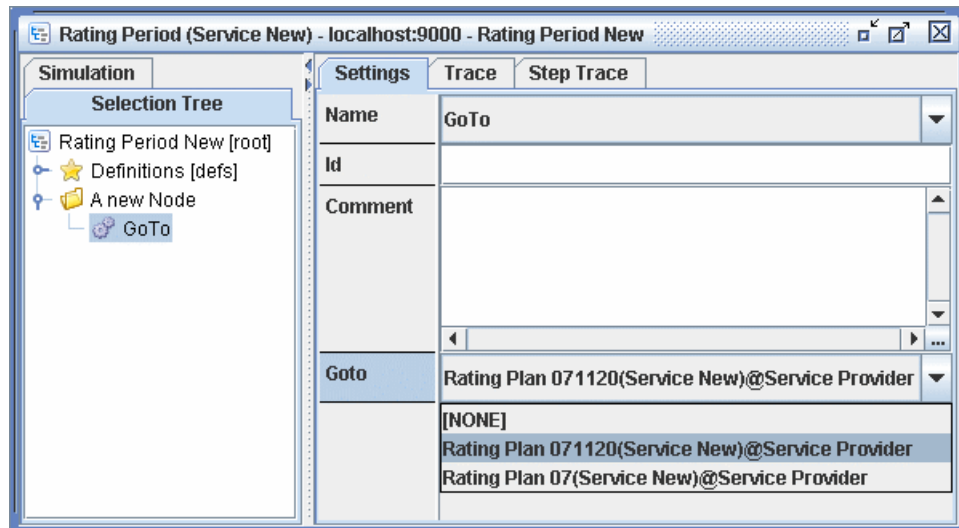


Figure 181 Selections for GoTo Modifier

Below the **GoTo** modifier is displayed, showing a jump in the Selection Tree, see Figure 182.

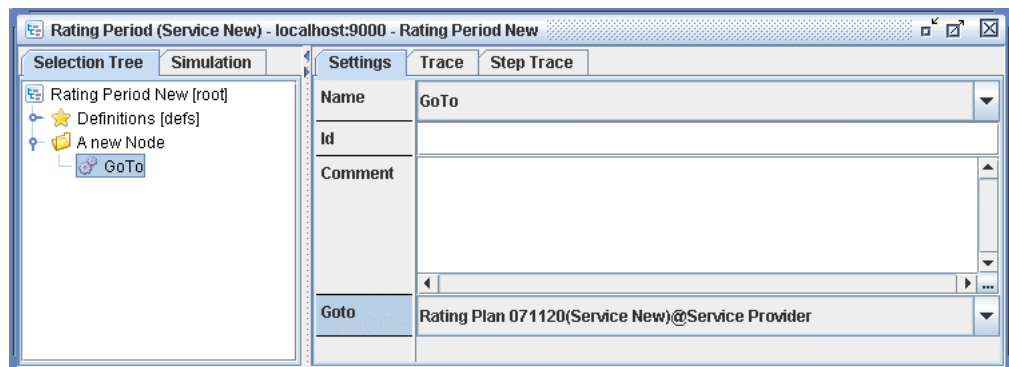


Figure 182 GoTo Modifier

If **NONE** is selected as a tree to go to, an exception is thrown in execution.

8.3.11

ClearValue

The **ClearValue** modifier enables the clearing of an associated field or a value in a value map. The supported parameter types are OUT and VARIABLE, see Figure 183

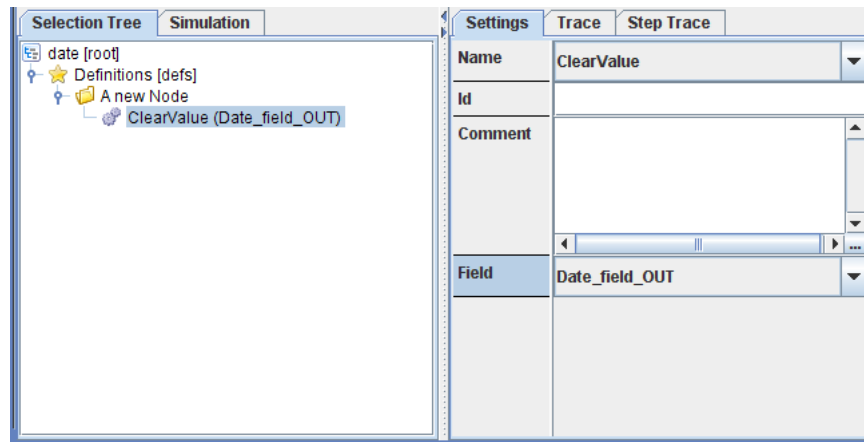


Figure 183 ClearValue Modifier

If no fields of the supported types have been created, the **Field** parameter shows **No available fields found**.

If the associated field is a map, a **Clear value for key** check box is visible. Selecting this check box allows a specific value in a map to be chosen as the field to be cleared. If the check box is not selected, then the map itself is cleared. By using constraints when setting up the service connected to the condition, it is possible to change the behavior of the clear value for key function. The solution that integrates ERE may already implement these changes. The following changes are possible:

- Hide the check box. This forces the map itself to be cleared, rather than a single value.
- Force the check box to be selected. This forces the user to specify the key to the map value to be cleared, rather than clearing the map itself.

For more information on how to use constraints, see Section 5.5 on page 60.

8.3.12 MultiFieldOperation

The **MultiFieldOperation** modifier makes it possible to perform an operation using several operands. The parameters from TDFs and SDFs are available for selection, see Figure 184. If the entered value is of the type **Amount**, then the operator has the behavior described in Section 8.3.12.1 on page 199. The result from the operation is stored in the **Destination Field** field.

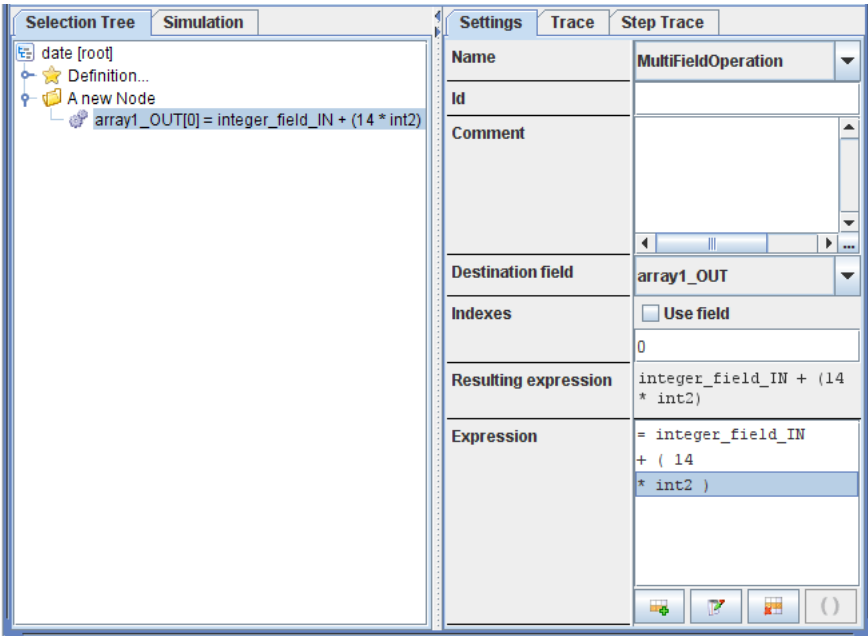


Figure 184 MultiFieldOperation Modifier

The mathematical expression to be calculated is defined row by row in the Expression field. A new row may be added using the button marked **Insert a new expression row below the selected row**. Existing expression rows may be editing using the button marked **Edit the selected expression row**. Rows may be removed from the expression using the button marked **Delete the selected expression row**.

Adding a new expression row or editing an existing row brings up a dialog in which the row is defined, see Figure 185. The operations available are found in Table 85.

Table 85 Operations Available in MultiFieldOperation

Operation.	<div>Add – adds a value to the expression.</div> <div>Subtract – subtracts a value from the expression.</div> <div>Multiply – multiplies the expression by a value.</div> <div>Divide – divides the expression by a value.</div> <div>Integer divide – divides the expression by a value, truncating the answer to the nearest integer. .</div> <div>Modulus – finds the modulus of the expression and a value.</div> <div>Power – raises the expression to the power of a value.</div>
------------	---

Note: The order in which the operations are carried out is parentheses, power, multiplication/division/modulus/integer division and addition/subtraction. The power operation is right associative, while the others are left associative.

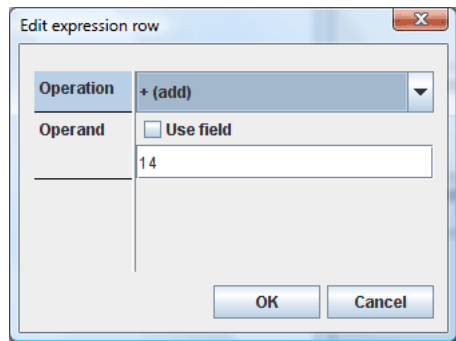



Figure 185 Edit Expression Row Dialog

The **Operand** may be defined in a number of ways. It may be entered manually, as a floating point number. It may be read from the service defined fields by selecting the **Use Field** check box. If the indicated field is a value list or map, it is possible to define the index of the value to be read from the list by entering the index number manually, or selecting the **Use Field** check box and choosing a field from the subsequent drop-down list.

When choosing more than one row of the expression, it is possible to use the parentheses button, , to set parentheses around those rows to further define the mathematical expression.

Several rows of the expression may be highlighted simultaneously.

When choosing more than one row of the expression, it is possible to use the parentheses button to set parentheses around those rows, to further define the expression.

Several mathematical functions have been defined, and may be selected using the **Add/insert a function button**. Selecting this button opens a drop-down menu with several predefined functions. The number of terms in the function is defined when adding the function, and may not be changed without removing the function and adding a new. The functions available are described in Table 86.

Table 86 Functions Available in Multi Field Operator

round (number)	Rounds the number to the nearest integer.
round (number, precision)	Rounds the number, with the number of digits after the decimal point defined by precision .
round (number, precision, mode)	Rounds the number, with the number of digits after the decimal point defined by precision . The different modes which may be chosen are: <ul style="list-style-type: none"> • 1 - Floor. • 2 - Ceiling. • 3 - Truncate. • 4 - Round. This is the default mode.



Table 86 Functions Available in Multi Field Operator

truncate (number)	Truncates the number to its integer part.
truncate (number, precision)	Truncates the number, with the number of digits after the decimal point defined by precision .
BalanceOf (amount)	Selects the numerical part of an amount.
BalanceOf (amount, currency)	Separates the numerical part of an amount from the currency.

Functions may be nested within one another. To nest a function, the expression row highest in the list is selected, and the button selected again. The most recently added function is placed closest to the operand.

To delete a function, the expression row highest in the list must be highlighted. Nested functions are deleted one at a time, starting with the last one added to the expression.


The **Resulting Expression** field displays the operation defined in the modifier, which also appears in the **Selection Tree** tab on the left.

8.3.12.1 Amounts in MultiFieldOperaton

It is only possible to use amounts in this modifier if the output field is defined as an amount. If the expression and the format of the output field are not compatible, an error is shown. Expressions that do not conform with the following rules will be marked red, and the tree cannot be saved.

- Amounts may not be added to or subtracted from a real number, and real numbers may not be subtracted from amounts.
- Amounts may not be multiplied by other amounts.
- An amount may only be divided by a real number. No number or amount may be divided by an amount.
- Adding or subtracting two amounts in different currencies uses the conversion factor linked to the service. The result will be in the currency which occurs first in the expression field.

8.3.13 DateOffset

The **DateOffset** modifier calculates the date resulting from adding or subtracting an offset in days to a defined date, and the result is output to a data field, defined in **Destination Field**. The date from which the offset is calculated may be read from the field defined in the data for the service by selecting the **Use Field** check box, and selecting the desired field from the drop-down list. The dates may also be entered manually in the format YYYY:MM:DD, or selected from the calendar by selecting the  button. The default value for the date is the current date.

The **Operation** field makes it possible to select if the offset should be added to or subtracted from the input date.

The **Resulting Expression** field displays the operation defined in the modifier, which also appears in the **Selection Tree** tab on the left.

The offset number of days may be defined in a number of ways. It may be entered manually, as an integer number of days. It may be read from the fields defined in the data for the service by selecting the **Use Field** check box. If the indicated field is a value list or map, it is possible to define the index of the value to be read from the list by entering the index number manually, or selecting the **Use Field** check box and choosing a field from the subsequent drop-down list.

Figure 186 DateOffset Modifier

Figure 186 gives an example of the Date Offset modifier. In this example, the value output to the destination field Integer_Field_OUT is 2009-08-20, the input date plus two days.

8.3.14

DateDifference

The **DateDifference** modifier outputs the difference in days between two dates to a data field, defined in **Destination Field**. The dates between which the difference is calculated may be read from the fields defined in the data for the service by selecting the **Use Field** checkbox, and selecting the desired field from the drop-down list. The dates may also be entered manually in the format YYYY:MM:DD, or selected from the calendar by selecting the button. The default value for the dates is the current date.

The **Resulting expression** field displays the operation defined in the modifier, which also appears in the **Selection Tree** tab on the left.

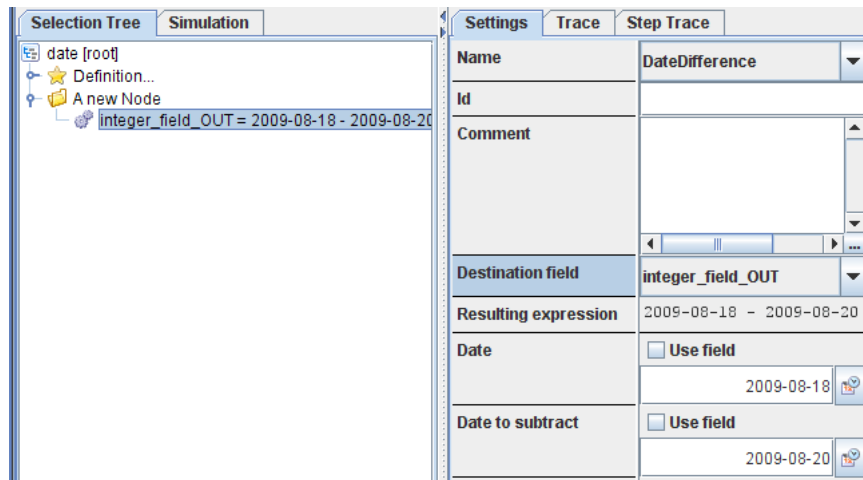


Figure 187 DateDifference Modifier

Figure 187 gives an example of the **DateDifference** modifier. In this example, the value output to the destination field Integer_Field_OUT is -2, the difference between the given dates.

8.3.15

BreakPoint

The **BreakPoint** modifier stops the selection tree execution, if the **Break** checkbox is selected.





9 ERE Tools

ERE provides tools to work with rating manager configuration and selection trees. These tools are described in this chapter.

9.1 Import Wizard

The import wizard provides the possibility to import rating manager configuration into the current navigator element.

The imported configuration is in the format of an XML file in a structure recognized by ERE. The files are generated by exporting rating manager configuration from another RMA using the export wizard, see Section 9.2 on page 204 for a description of this tool.

The types of content that may be imported are:

- An entire rating manager configuration.
- Navigator elements, such as a service or a rating plan.
- Selection trees.
- TDFs, only available when importing to the Tree Defined Fields editor. See Section 6.6.2 on page 79 for more information on the editor.

When importing content to the current configuration, the user is prompted to choose if the imported and existing configuration are to be merged if some of the structures imported already exist in the current configuration. Choosing **Stop** cancels the import. Choosing **Yes** merges the two configurations, where the user can choose which structures are to be merged. The user is also prompted if such things as start dates conflict and so can manually change them.

It is possible to force the import of a service definition that is not consistent, but this is not recommended.

The application that integrates ERE may set restrictions on the content that may be imported. A warning will be shown and the import action canceled

The wizard may be opened in several different ways, depending on where in RMA the user is currently working. The different ways are:

- Selecting **Import** from the rating manager menu.
- While having the desired element active, select the **Import from file** icon from the RMA toolbar.
- While having the desired element active, press `ctrl + I`.
- Drag and drop content to a navigator element or a service editor.

9.1.1 Steps in the import wizard

1. Open the wizard using one of the methods described above. Browse to and select the XML file. Once a file has been selected, the wizard tries to validate its contents to make sure that it is a properly formatted file. If the import file does not have all the configuration information needed, the file may be migrated to the proper format. The missing information may be added in the dialog that opens.
2. The configuration in the selected file that is available for import is shown as a tree structure, select the parts to be imported, see Figure 188.
3. Select **Next**. Information on what has been imported is seen in the log in the dialog.

Note: Once the import process has started it is irreversible, that is there is no way to undo the import as it progresses. Pressing the stop button stops the import but it does not roll back the navigator elements that have already been imported.

4. Select **Finish**.

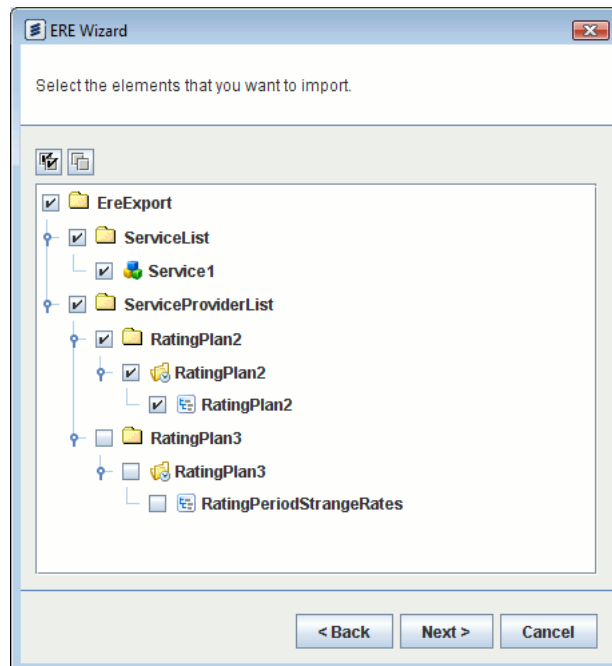


Figure 188 Import Wizard, Select Nodes

9.2 Export Wizard

The export wizard provides the possibility to export rating manager configuration from ERE to a file.



The content exported consists of rating manager configuration in the form of an XML file. Configuration exported may be imported by other RMA instances.

The types of content that may be exported are:

- An entire rating manager configuration.
- Navigator elements.
- Entire selection trees.
- TDFs, only available when opening the tool from the root of a selection tree.

The options of what may be exported may depend on the solution in which ERE is integrated. See the documentation that accompanies the solution for more detail.

The wizard may be opened in several different ways, depending on where in RMA the user is currently working. The different ways are:

- Selecting **Export** from the rating manager menu.
- While having the desired element active, select the **Export to file** icon from the RMA toolbar.
- While having the desired element active, press `ctrl + E`.

9.2.1 Steps in the Exporting Wizard

1. Highlight the element to be exported from the navigator, and open the wizard.
2. Browse to the directory which will contain the export file, and name the export file. If an existing file is selected the wizard prompts for confirmation before proceeding the export. Once a file has been selected, press **Next**.
3. Choose the rating manager configuration to be exported. If the export was initiated from a leaf node in the navigator tree, only that node is presented in the wizard. If the export was initiated from a higher node in the navigator tree, it is possible to select or deselect the configuration to be exported, see Figure 189.
4. Press **Next** to go to the next step. This gives an opportunity to change additional options for the export, which may depend on the solution in which ERE is implemented.
5. Press **Next** to start the export and generate the export file. While the export is progressing the **Cancel** button can be used to stop the export process. Information on what has been exported is seen in the log in the dialog.

Note: Review the log for any warnings or errors before closing the wizard.
6. Select **Finish**.

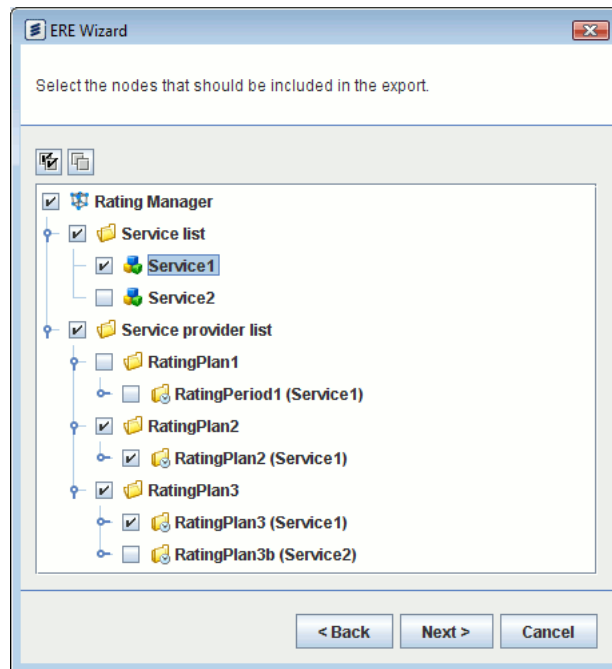


Figure 189 Selecting Configuration for Export

9.3 Distribution Wizard

The Distribution Wizard is a tool for managing services and rating period configuration across multiple ERE instances available in the navigator. Services or selections trees can also be deleted from several ERE instances simultaneously. If a selection tree is to be distributed to an ERE instance which has no matching rating plan and service provider, these are automatically created.

Distributing configuration to ERE instances which contain elements with the same name will overwrite the content in the destination ERE instance.

The wizard analyzes if it is possible to do the distribution before starting. This information is shown in the panel under each ERE instance in the wizard.

The wizard may be opened in several different ways, depending on where in RMA the user is currently working. The different ways are:

- While having the desired service definition or rating period selected in the navigator window, choose **Distribute...** from the rating manager menu.
- While having the desired service definition or rating period selected in the navigator window, choose **Delete from several EREs** from the navigator window
- When working in the selection tree editor, choosing **Save to multiple EREs**.



In the wizard, the ERE instance on which the selected service or rating period is configured is called an Associated ERE. All other ERE instances in the navigator are Other EREs.

The actions that are available in the wizard may depend on the solution in which ERE is implemented. See the documentation that accompanies the solution for more detail.

9.3.1 Steps in the Distribution Wizard

1. Highlight the element to be sent or deleted, and open the wizard. Select at least one ERE which the configuration is to be distributed to or deleted from. The wizard checks to see which ERE instances the configuration can be distributed to or deleted from. If none exist, it is not possible to proceed.
2. Check the color of the ERE instance name, and the expandable information panel under each ERE instance, see Figure 190. The available colors are:

- Green - The task may be performed.
- Yellow - The task may be performed, but may not be advisable. Further information is shown in the panel.
- Red - The task may not be performed.

There are several options available when viewing the ERE instances in the wizard. They are:

- **Select All** - All the ERE instance listed in the wizard are selected.
- **Deselect All** - Clears all currently selected ERE instances.
- **Expand All** - The entire information panels are displayed.
- **Collapse All** - Minimizes the information panels.

3. Select at least one ERE instance and click **Next**. Information on what has been distributed is seen in the information panels for the ERE instances.

Note: Review the information panels for any warnings or errors before closing the wizard.

If the task is unsuccessful for some of the ERE instances, the distribute task is carried out on the rest of the ERE instances. It is possible to select **Back**, and return to the previous step. The user is prompted if they want to analyze the selected ERE instances again.

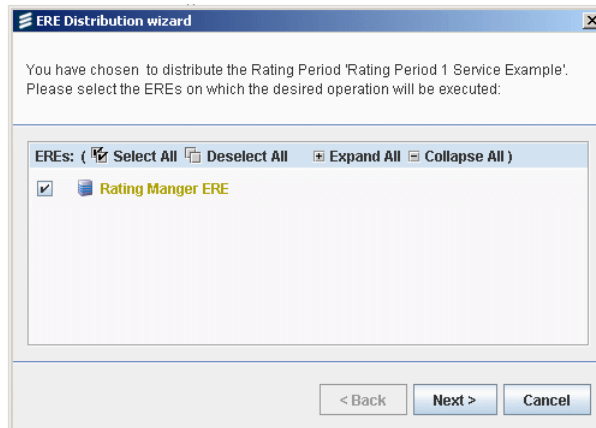


Figure 190 An Example of ERE Distribution Wizard

9.4 Selection Tree Difference Detection Tool

The selection tree difference detection tool, also called the diff tool in this document, allows two selection trees to be compared to one another, highlights the differences and controls the merging of the rating periods.

The rating periods to be compared are displayed beside each other in the window of the tool.

The names of the nodes and sub-nodes that contain differences are highlighted as colored text in the trees in the window. Selecting a node that contains a difference opens a tab in the window that contains a description of what differs between the two rating periods.

In the selection tree difference detection tool, the first tree is taken as a reference for comparison with the second tree. The types of difference that are detected by the tool are as follows:

- **Children differ:** a sub-node, modifier or condition occurs in one selection tree, and not in the other. This is highlighted using green text for the node name of the parent node.
- **Child order differs:** the order of sub-nodes, conditions and modifiers in a node of one selection tree differs from the other. This is highlighted using yellow text for the node name of the parent node.
- **Parameters:** The parameters, such as the node name, differ between corresponding nodes in the selection trees. This is highlighted using blue text for the node name of the parent node.

The tool can be opened in several ways. They are:

- From the **Util** tab in the **Main** menu, open the tool. Drag-and-drop the rating periods from the navigator to the tool window.



- Select the **Show Diff Tool** icon from the RMA toolbar. Drag-and-drop the rating periods from the navigator to the tool window.
- In the navigator, right-click on a rating period, and chose **Compare to** Drag and drop the second period into the tool window.
- If another user has edited and changed the rating period while the current user has had it open, the current user is prompted to choose to overwrite the saved version, accept the saved version, or cancel the operation. The first two options open the tool.

The tool window is described in Figure 191 and Table 87.

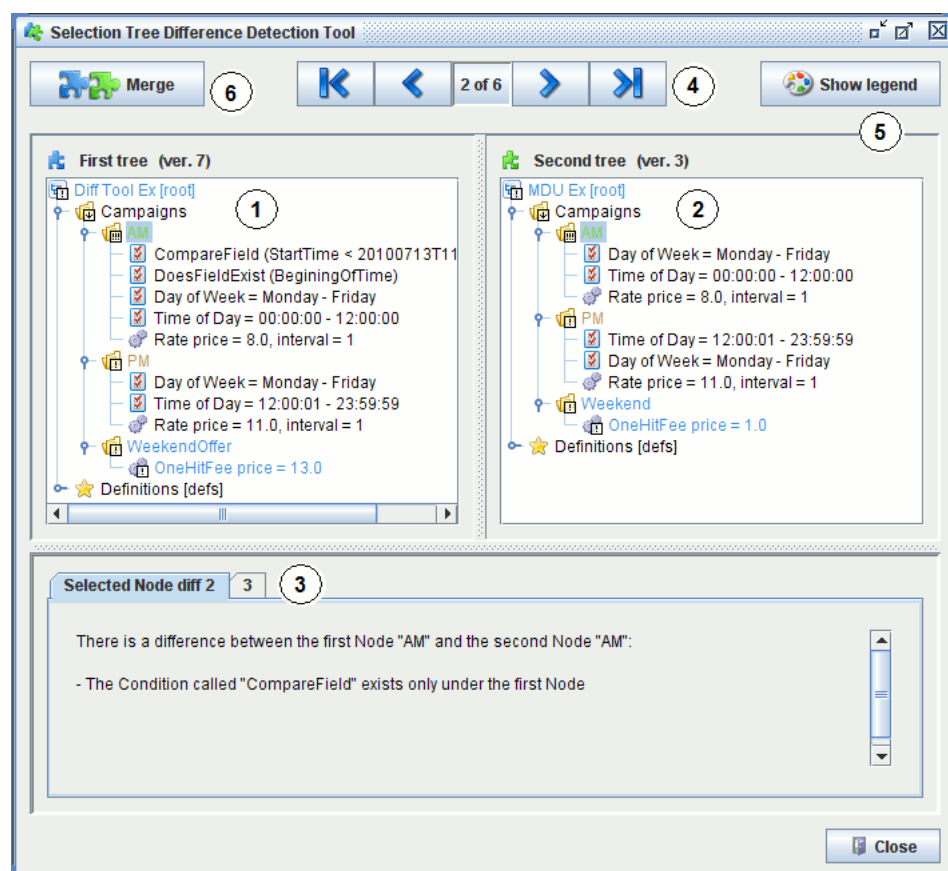


Figure 191 The Selection Tree Difference Detection Tool

Table 87 Legend to the Diff Tool Figure

1	First Tree	The first tree selected. Also called the local tree, depending on how the tool is opened.
2	Second Tree	The second tree selected. Also called the remote tree, depending on how the tool is opened.

Table 87 Legend to the Diff Tool Figure

3	Difference description tabs	Describes the difference between the two selection trees open in the window. There is one tab visible for each difference existing in the currently selected node.
4	Difference navigator buttons	Scrolls through the differences detected between the two trees.
5	Legend	Opens the legend panel, described in Figure 192.
6	Merge	Opens the merge view to start the merge process. See Figure 193.

The legend panel helps the user to interpret the icons in the tool. It is described in Figure 192 and Table 88.

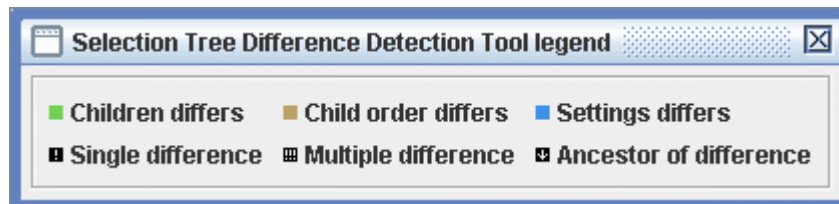


Figure 192 Legend Panel

Table 88 Legend to the Legend Panel

Children differs	A sub-node, modifier or condition occurs in one selection tree, and not in the other.
Child order differs	The order of sub-nodes, conditions and modifiers in a node of one selection tree differs from the other.
Settings differs	The parameters, such as the node name, differ between corresponding nodes in the selection trees.
Single difference	The nodes have only one difference between the trees.
Multiple difference	The nodes have more than one difference between the trees.
Ancestor of difference	One or more of the sub-nodes contain a difference.

The merge process controls which of the two selection tree configurations are saved, one difference at a time. By default, it is the behavior of the first/local tree that is chosen.

Choosing to **Restart merge** discards the resulting tree changes, and runs the analysis again.

The tool window during the merge process is described in Figure 193 and Table 89.

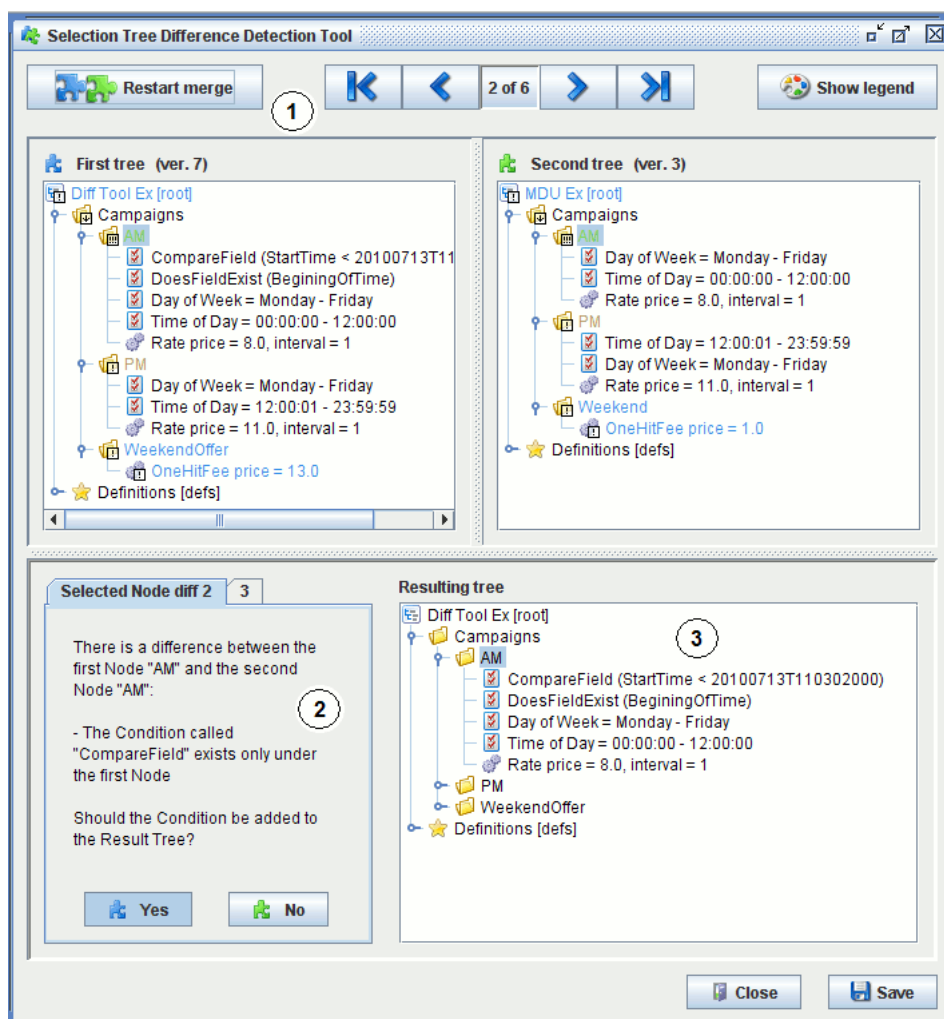


Figure 193 Merging with the Diff Tool

Table 89 Legend to the Merge Figure

1	Restart merge	Discard changes and restart the merge process.
2	Merge decision tab	Choose if the configuration for the first or second tree will be saved, for the difference in question.
3	Resulting tree	The tree that results from the merge process, ready to be saved.

When saving the resulting selection tree, there are several options available. The selection tree may be exported, similar to the method used in the **Export Wizard**, see Section 9.2 on page 204.

The resulting selection tree may be saved to the current ERE instance, overwriting either the first or second selection tree or both. The overwritten periods may also be saved to a backup folder.

9.5 Find Function

The find function is used to quickly find selection tree items in the Selection Tree Editor, which is useful for finding certain elements in large trees without the need of browsing through the whole tree. The function is accessed by using the **Find** option in the Selection Tree Editor context-sensitive menu. It is also possible to access it by pressing **CTRL+F**.

The find function window has two tabs. The first tab displays the **Quick** dialog and the second tab displays the **Advanced** dialog.

The quick search window is displayed as in Figure 194 and further described in Section 9.5.1 on page 212.

The advanced search window is displayed as in Figure 195 and further described in Section 9.5.2 on page 212.

9.5.1 Quick

To perform a quick search, type in one or several search criteria in the search field and click the Find-button. If several search criteria are included, all must be fulfilled for an item in the tree to be found. The search term 'OR ' may be used to refine the search. If 'OR' is used, it is enough that one of the search criteria is fulfilled for an item in the tree to be found.

Example: If the words “week time” are entered in the search field, the result only includes tree items with both these words. If “OR” is included in the search criteria, “week OR time” here, all items containing the words “week” or “time” will be returned.

The quick search function always starts at the root of the tree and is not case-sensitive. The search result is displayed in the area below the Find-button, as in Figure 194.

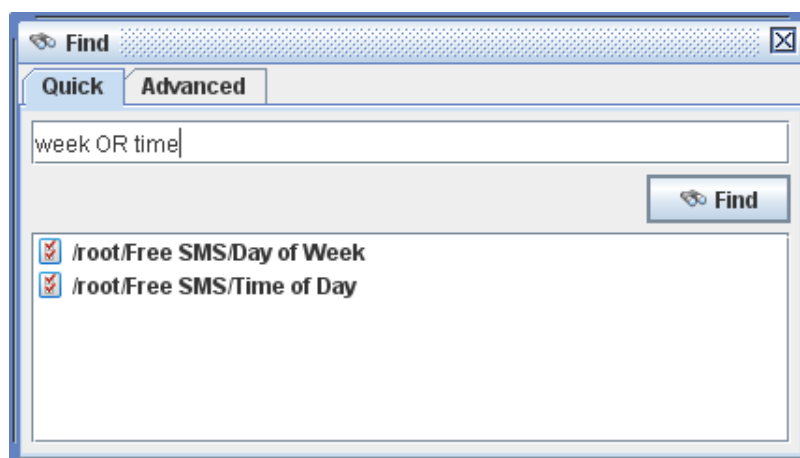


Figure 194 Quick Find



9.5.2

Advanced

The advanced search function may be used to search for the following:

- Name or Id
- Parameter
- Ref. Count
- Tag
- Has Tag
- Qualifier
- Uses TDF

The criteria used for the advanced searched is further explained in Table 92.

The search result is displayed in the area below the Find-button, as in Figure 195.

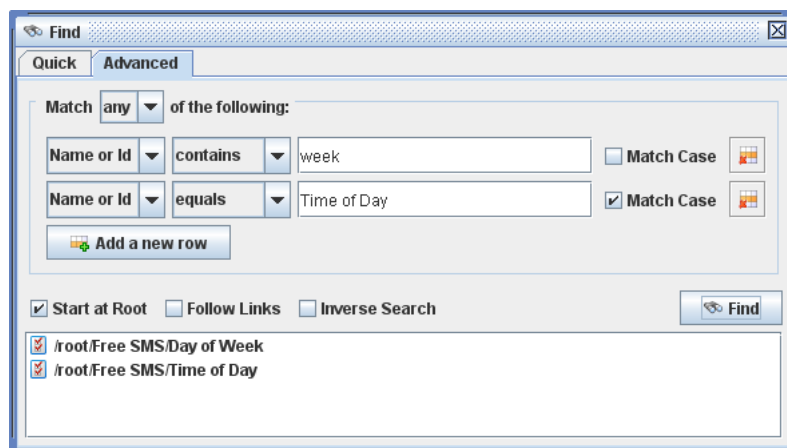


Figure 195 Advanced Find

The **Advanced** search criteria can be combined with logical search functions, Table 90 describes the options in the **Search Criteria** drop-down menu in the advanced find function.



Table 90 Advanced Search Logic

Drop-down menu	Description	Format
Match of the following	Drop-down menu describing the search logic if more than one search criteria exist.	<p>The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none">• All – functions as a logical AND which means that all search criteria listed must be fulfilled.• Any – functions as a logical OR and means that any of the listed search criteria can be fulfilled for the search to be successful and displayed in the search output window.

Table 91 Description of the Advanced Find Window

	Description
Search criteria	<p>Search criteria check box.</p> <p>Check box checked – Searches for entries with that match the case of the search entry.</p> <p>Check box cleared – Searches for entries which match the search entry without being case-sensitive.</p>
Match Case	<p>Search criteria check box.</p> <p>Check box checked – Searches for entries with that match the case of the search entry.</p> <p>Check box cleared – Searches for entries which match the search entry without being case-sensitive.</p>
Add a new row	<p>Search criteria button.</p> <p>There must always be at least one search criteria displayed.</p> <p>A maximum of six rows may be selected.</p>
Remove this row	<p>Search criteria button.</p> <p>Removes the corresponding row from the search criteria.</p> <p>There must always be at least one search criteria displayed.</p>
Follow links	<p>Search criteria check box.</p> <p>Check box checked – Returns both the selection tree elements that contain a link, and the targets of these links, for example in the definitions node.</p> <p>Check box cleared – Returns only the elements.</p>



Table 91 Description of the Advanced Find Window

	Description
Start at root	Search criteria check box. Check box checked – Starts the search at the selection tree root. Check box cleared – Starts the search from the selected item in the selection tree.
Inverse Search	Search criteria check box. Check box checked – Searches for entries that do not match the criteria. Check box cleared – The resulting search output matches the input criteria.
Find	Search criteria button. Press this button to execute the search.
Search output	Search result window. Displays the result of the search. Click on an item displayed to jump to that location in the selection tree.

Table 92 Advanced Search Criteria Row

Search Criteria Row	Format
Name or Id – search only for the name or ID of a node, condition or modifier.	<p>Name or ID evaluation drop-down menu. The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none"> • equals • contains • starts with • ends with <p>Name or ID input field – text field where the name or ID of the selection tree element can be entered. This field is compared against the values for the Name and Id fields shown in the Settings tab in the selection tree element.</p> <p>For an example of how to use the Parameter search criteria, see Section 9.5.3 on page 217.</p>



Table 92 Advanced Search Criteria Row

Search Criteria Row	Format
Parameter – search conditions and modifiers for parameters which matches the search criteria.	<p>The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none">• equals• not equals• is greater than• is less than• contains• starts with• ends with <p>For an example of how to use the Parameter search criteria, see Section 9.5.4 on page 218.</p>
Ref. Count – search for nodes, conditions or modifiers that are referenced a certain number of times using the linking function.	<p>Reference count evaluation drop-down menu. The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none">• equals• not equals• is greater than• is less than <p>Reference count value parameter – a numerical value of the number of times a link is used.</p> <p>Note: If the option Equals 0, is entered the search criteria lists all nodes, conditions, and modifiers that are not linked.</p>
Tag – search only for the tag of a node, condition or modifier.	<p>Tag evaluation drop-down menu. The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none">• equals• contains• starts with• ends with <p>Tag input field – text field where tag number of the selection tree element can be entered. This field is compared against the values for the Tag field shown in the Settings tab in the selection tree element.</p>



Table 92 Advanced Search Criteria Row

Search Criteria Row	Format
Has Tag - search for all selection tree elements that have a specified tag.	
Qualifier – search for nodes, conditions, or modifiers that have an STQ which matches the search criteria.	<p>This criteria is only available when STQs have been defined in the selection tree.</p> <p>Qualifier evaluation drop-down menu. The options in the drop-down menu are as follows:</p> <ul style="list-style-type: none"> • equals • contains • starts with • ends with <p>Qualifier input field – text field where STQ number of the selection tree element can be entered. This field is compared against the values for the Qualifier field shown in the Settings tab in the selection tree element.</p>
Has Qualifier - finds all selection tree elements that contain an STQ.	
Has Shared Qual. - finds all nodes that have the same STQ.	
Uses TDF - search for nodes, conditions or modifiers that have a TDF which matches the search criteria.	

9.5.3 Finding All Modifiers of a Certain Type

In this section, an example of how to find modifiers of a certain type is described, see the following step-list.

This example shows how to find all Set field modifiers in the selection tree.

1. Select **Find** in the context-sensitive menu in the tree, or press **CTRL+F**.

2. Choose **all** in the drop-down. This makes the search match all the criteria specified below.
3. Specify the search criteria. In this case we want to find all Set field modifiers. Select **Name or Id** in the first drop-down.
4. Choose **contains** in the second drop-down.
5. Write **Set field** in the Text field. This makes up one search criteria.
6. If the user did not stand in the root of the tree when Find was selected, the check box **Start at root** should be checked if the find function should search the tree from root to bottom. Otherwise, the search is executed from the point in the tree where the Find function was selected.
7. Press **Find**.

In this case all Set field modifiers and the path to them in the tree (root and nodes), is displayed in the Search output.

9.5.4 Finding Parameters in Selection Tree Items

In this section an example of how to find parameters, containing certain values, in selection tree items. It is described in the following step-list.

This example shows how to find specific values in parameters of a condition or modifier. In this case, the user wants to find the numbers 100, 245 and 900.

1. Select **Find** in the context-sensitive menu in the tree or press **CTRL+F**.
2. Choose **any** in the drop-down. This makes the search match any of the criteria specified below and enables the user to find items in the tree where the parameters can contain any of the numbers 100, 245 and 900 or all of them.
3. Specify the search criteria. In this case we want to find all parameters containing a **Numbers** parameter that has one or all of the values 100, 245, 900.
4. Select **Parameter** in the first drop-down.
5. Enter the text **Numbers** in the text field.
6. Choose **contains** in the second drop-down.
7. Enter **900** in the text field. This makes up one search criteria.
8. Add another search criteria by pressing the **+** button.
9. Enter the value **100** in the last text field.
10. Do the same with the last search criteria, but enter the value **245**.
11. If the user did not stand in the root of the tree when Find was selected, the check box **Start at root** should be checked if the find function should search



the tree from root to bottom. Otherwise, the search is executed from the point in the tree where the Find function was selected.

12. Press the **Find** button. In this case, all conditions and modifiers with the Parameter Numbers that contains 100 and/or 245 and/or 900, and the path to them in the tree (root and nodes), is displayed in the Search output.

9.6 Branch Search

Branch search is an advanced search function which finds branches in selection trees. It makes it easier to find branches with a certain layout, and present them in a table form. It also makes it easier to find selection tree elements, depending on how they are defined. Branches are entire sections of a selection tree from the root of the tree to a leaf element, that is, a node that contains a modifier as its last element. Branch search allows the user to configure one or more filters, and use these filters to search through a selection tree for branches that fill certain criteria. The search is defined by one or several filters, and several filters may be combined to create complex search criteria.

The branch search is started from the selection tree editor, by choosing the menu option **Search for branches....** Each filter created must have a unique name. The branch search dialog is described in Figure 196 and Table 93.

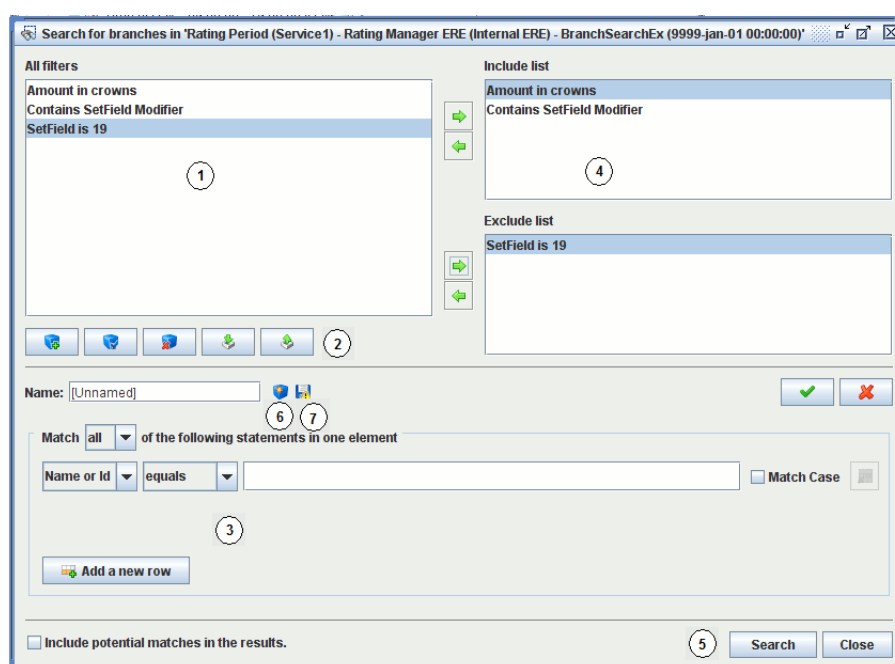


Figure 196 Branch Search Dialog with Some Examples of Filters

Table 93 Legend to Branch Search Figure

1	List of all filters available for the current service.
2	Buttons to create and administer filters.



Table 93 Legend to Branch Search Figure

3	Fields to create and edit filters.
4	Lists of included and excluded filters for the current branch search.
5	Button to start the branch search.
6	Filter information - the filter being edited is a new filter.
7	Filter information - the filter definition has been changed.

Two types of filter may be defined - **Field Value** and **Element Setting** filters. Both types define criteria that are to be matched to all branches in a selection tree. The filters are connected to the service definition associated with the tree, and are saved automatically per service.

Field Value filters are set up using the fields in the service definition associated with the selection tree. A data value to be matched against may also be provided to the selection tree, so that the resulting behavior is also included in the search. Several fields may be added to a filter to narrow down the results. Once a field from a specific context in the service is used, only other fields in that context, or belonging to no context, may be chosen in the same filter. The filter judges the relevance of branches, the more criteria a branch fills, the higher its relevance in the search result.

Values for the field may be provided in field value filters. If no values are supplied, then the default value of the field in the service definition is used. The conditions in the selection tree are executed, so that the results returned are used as the target of the search.

Branches that match the search criteria defined are returned in the results. Branches that may match the criteria, but cannot be evaluated in a field-value filter because of no conditions, are known as potential matches. They can be added to the search results by selecting the check box in the branch search window.

Element Setting filters are set up using the name or ID, the value of a certain parameter, or STQ of an element or elements in a branch of the selection tree. Up to three of these criteria may be used in the same filter. The filter may be configured to use a logical AND by selecting **Match all**, or a logical OR by selecting **Match any**, to combine these search criteria.

Element setting filters match the criteria in string form or floating point value form with the settings of the elements in a selection tree. No wildcards may be used in the filter definition.

A filter defined must be saved before they can be used in a branch search. The filters are saved to the `.rma` folder for the current user. Removing a filter from the All Filter list in the branch search dialog removes this filter from the `.rma` folder, meaning that it cannot be recovered. Filters defined for one selection tree may be used in other selection trees that share the same service definition.



When performing the search, filters may be included or excluded. Excluded means purposefully eliminating the terms in that filter from the search results, to remove irrelevant matches from the result. Branch search first searches the tree using the filters in the include list, and then applies the filters in the exclude list to remove irrelevant results.

If no filters are defined, performing the search will display all branches in the selection tree.

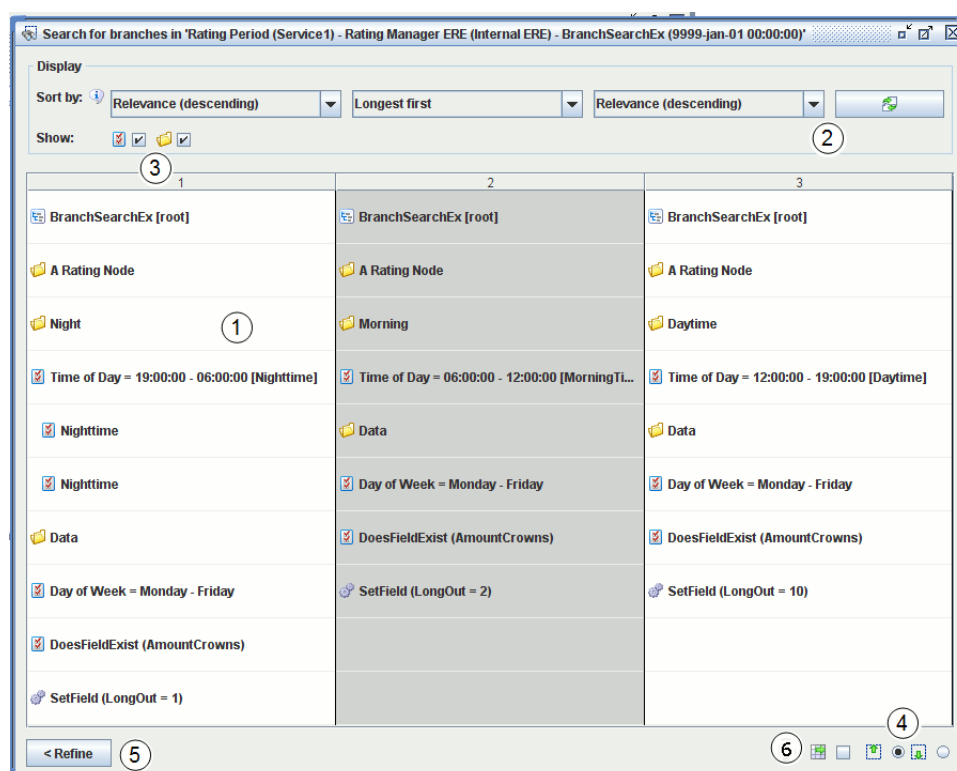


Figure 197 An Example of the Results of a Branch Search

Table 94 Legend to Branch Search Results Figure

1	The results of a branch search. Each column is a separate branch of the selection tree.
2	Criteria to sort the search results by.
3	The buttons to hide or show conditions and nodes.
4	Buttons to toggle if the branches are shown with root first or last in the column.
5	Return to the first page and refine the branch search.
6	Button to toggle if each branch should be displayed as a column or a row.

The results of the branch search are presented in a table, the form of which may be configured. Figure 197 and Table 94 show the result table with an example of the results of a search.



If no branches in the selection tree are returned by the search, **No matching branches found** is displayed.

Each column of the results table represents a single branch of the selection tree, from root to leaf. Checking the **Show branches in row mode** box rearranges the table so that each row is a single branch. Double-clicking on a cell in a branch brings up the same element in the selection tree editor. The branches may be shown in two ways, either with the root of the tree at the top or at the bottom of a column. This is controlled by the **Toggle showing root as top node** and **Toggle showing leaf as top node** option buttons.

To present more concise branches in the table, it is possible to hide all conditions, nodes, or both in the table, using the **Show:** check boxes.

The branches shown in the result table may also be sorted to present the most interesting results first. Up to three sort criteria may be used. The **Arrange columns by selected criterion** button sorts the results table according to these. These criteria may be:

- **Relevance** - The branches that meet most search criteria are shown first.
- **Location in tree** - Those results that occur closest to the root in the selection tree editor are displayed first.
- **Longest first** - The branches with most visible elements are displayed first.
- **Shortest first** - The branches with the fewest visible elements are displayed first.
- **Settings** - Brings up a dialog, where the parameter to sort by may be defined. The setting can be any of the parameters for any of the elements in the result. It is possible to display the branch in ascending or descending order of the value of the parameter chosen. These settings only take affect when **Sort** is selected.

The **Refine** button returns to the dialog where the filters are defined, so that further criteria may be added to the search.

Closing a selection tree editor while an associated branch search editor is open, brings up a dialog asking if both editors should be closed. Selecting **No** leaves both editors open.

9.7 Multi Data Update Tool

The Multi Data Update (MDU) tool enables related data to be updated in several places in a selection tree at once. It allows selection tree elements to be found in a selection tree, based on criteria entered in the MDU window, and selectively update the data in the relevant elements. This helps separate the data in the selection tree from the logic. It simplifies the updating of data in large selection trees; for example, call prices which can change quite regularly in many elements of a tree.



New data values can be applied to the selection tree elements that match the criteria are returned from the search. The resulting data is then loaded back in to the selection tree.

The data may be imported from or exported to Comma Separated Value (CSV) files which may be read and edited in an external spreadsheet program.

Not all conditions and modifiers have parameters values that may be updated by MDU. Only the selection tree elements that currently support the tool have the possibility to set the **Tag** parameter. When defining the criteria for finding elements containing data, it is recommended to filter by **Tag**, as the filter by **Path** works best in trees with a well defined structure.

9.7.1 Using the MDU Tool

The Multi Data Update tool window may be opened in several different ways. These are:

- With the rating period containing the selection tree to be updated selected in the ERE navigator, select **Multi Data Update** from the rating manager menu.
- While having the rating period containing the desired selection tree selected in the ERE navigator, press **ctrl + M**.

The MDU tool window consists of a message panel, a panel in which the search criteria for the data to be updated is entered and result tabs that show the data returned by the search and in which the data may be updated. Figure 198 shows an example using the tool.

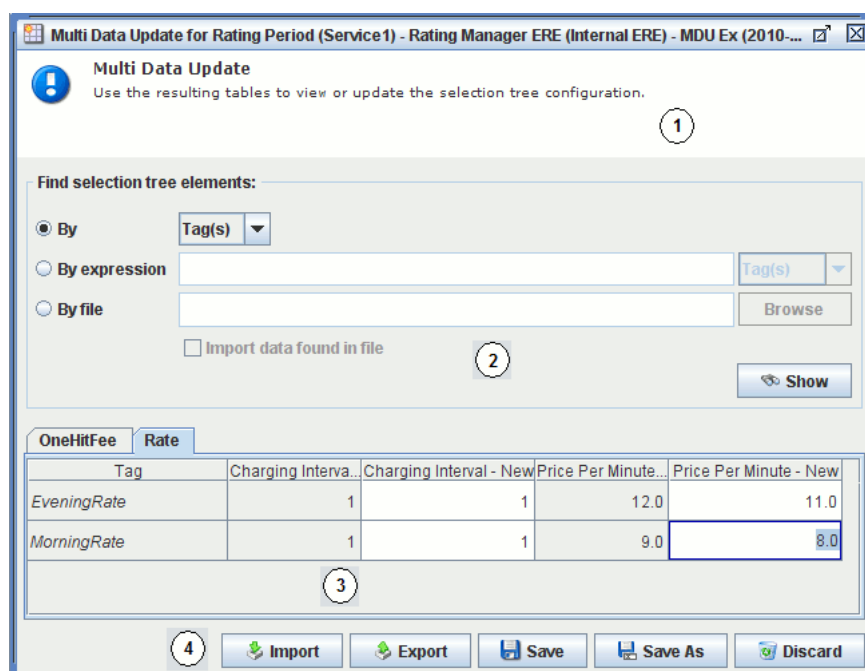


Figure 198 An Example of the MDU Tool.



Table 95 Legend to Fig 198

1	Message panel.
2	Filter criteria are defined here.
3	Result tabs.
4	Action buttons.

The message panel shows error messages and warnings, informs the user of the result of their previous action and gives tips on corrective action if the action has failed. Some of the error messages and warnings are described in Table 98.

The panel **Filter selection tree elements based on:** is where the criteria for finding the data to be updated is defined. The data may be searched for in three ways. They are:

- **By.** All selection tree elements containing a tag are returned when **Tag(s)** is chosen. All selection tree elements that support MDU are returned, along with the path to these elements, when **Path(s)** is chosen.
- **By expression.** The search criteria are defined using the paths or tags of the elements in the selection tree. The criteria take the form of lists of tags or paths, separated by a space. Some examples of how the criteria may be formulated, and the expected results are shown in Table 96.
- **By file.** A compatible file may be imported, creating a filter based on either paths or tags of selection tree elements. If the **Import data found in file** check box is chosen, both the filter and the data values in the file are imported.

Table 96 Common Criteria Used in the Filter by Field

Description of Filter	Criteria	Results in
Several elements with tags, separated by a space.	Tag1 Tag2	Elements with the specific tags Tag1 and Tag2.
An element with a space in its tag or node name.	"A descriptive text"	The elements with the tag "A descriptive text".
Elements, the path of which is known.	/node/modifier	The elements with the specified path. The root node of the tree is not needed when defining a path.
Elements whose tag parameter begins with "Tag". In other words the character '*' is a wildcard.	Tag*	All elements with a tag name beginning with "Tag".
Elements whose tag parameter ends with "Tag"	*Tag	All elements with a tag name ending with "Tag".
Elements whose tag parameter contains "Tag"	*Tag*	All elements with a tag name containing "Tag".



Table 96 Common Criteria Used in the Filter by Field

Description of Filter	Criteria	Results in
Elements, where the start of the path is known.	/node/**/	The elements with a path starting at the specified node. The root node of the tree is not needed when defining a path.
Elements, where the start of the path and the elements name is known.	/node/**/modifier	The elements with the name "modifier", with a path starting at the specified node.
If the element should have the character '*' in its path or tag, this combination is not read as a wildcard.	*	Allows the definition of an expression containing the character '*' as a part of the path or tag. In other words, '*' is not considered as a wildcard.

The filter is applied to the selection tree when the **Show** button is selected.

The **Result** panel displays the elements returned by the filters. Each element type returned has a separate tab in the panel. Each tab is presented as a table. The columns are explained in Table 97.

Table 97 MDU Result Panel

Column	Description
Tag or Path	<p>The tag name for the selection tree elements returned by the filter, if the filter criteria are based on tags. If several selection tree elements have the same tag name, the row in the table is marked with an (insert icon pic here) icon. All elements with the same tag will be updated at the same time.</p> <p>The path name for the selection tree elements returned by the filter, if the filter criteria are based on paths. If several selection tree elements have the same path, the row in the table is marked with an (insert icon pic here) icon.</p>
Old value	<p>The current values of the data from the selection tree.</p> <p>If there are more than one unique value for selection tree elements with the same tag, a mouse-over tooltip shows the values of the different values.</p>
New value	<p>The new values for the data for the selection tree. The values may be entered manually or imported from a file. They do not take effect on the selection tree until the Save button is selected, and the action is confirmed from the subsequent dialog.</p> <p>Right-clicking on the new values column gives the option to restore the values from the old value column.</p>

It is possible to copy the values from a result tab to the computers clipboard. These may then be pasted to a spreadsheet program. Likewise, it is possible to paste the values from a spreadsheet into a result tab. The table in the result tab and the spreadsheet program must have the same format for this to work, this is ensured by only using files or clipboard data which have been exported from the MDU.

If the data in the **New values** have been edited without being saved, and an attempt is made to paste in values from the clipboard, a warning similar to that in Figure 199 is shown. Choosing **Yes** overwrites the values in the columns with the values from the clipboard. Choosing **No** stops the paste, and retains the values in the **New values** columns.

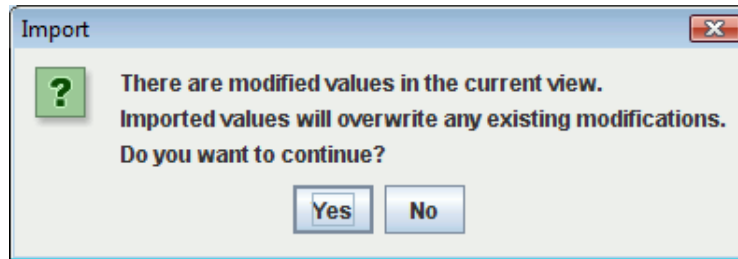


Figure 199 Warning Message when Importing MDU Values

There are a number of buttons which perform actions using the values in the results tab. These are:

- **Import.** This imports a CSV file containing a table of values to the current result tab. The layout of the values in the import file must be identical to the table in the results tab. One way to ensure this is to create the table in the MDU tool and export it, editing only the values later in a spreadsheet program. The external spreadsheet program may impose formatting on the CSV file when importing the data into cells. To avoid this, the cells in the spreadsheet should be formatted as text.

If the data in the **New values** have been edited without being saved, and the import function is started, the warning in Figure 199 is shown. Choosing **Yes** overwrites the values in the columns with the values in the imported file. Choosing **No** stops the import, and retains the values in the **New values** columns.

When importing a CSV file, some error messages and warnings may be displayed in the message panel if the file has an incorrect layout, or values are in the wrong format. These are described in table Table 98.

- **Export.** This exports the values in the **Old values** column of the result tab to a CSV file, which may be edited in an external spreadsheet program or imported into another RMA instance. If a row contains several values, only the first value shown in the tooltip is exported.
- **Save.** This saves the values in the **New values** column to the current selection tree if new values of the data have been entered. Check the message panel to see if any errors occurred while saving the new data.
- **Save As.** Save the values in the **New values** column to a new selection tree. The new tree has the same logic as the current selection tree. The new selection tree may be compared to the old tree using the selection tree difference detection tool, see Section 9.4 on page 208.



- **Discard.** Clears the result tabs, and all the values in the columns, without affected the values in the current selection tree.

Table 98 Error Messages and Warnings in MDU Tool.

Message	Severity	Scenario	Meaning
Could not read from file. File not found.	Error	When trying to find selection tree elements by file.	Selected file does not exist on provided path.
Please consult the user guide for information on pasting data in table.	Error	When pasting into MDU from system clipboard.	Data being pasted in has the wrong format.
Failed to paste data from clipboard", "An unexpected error occurred. See log entry [logentry] for more information.	Error	When pasting into MDU from system clipboard.	Data has a format not supported by MDU, for example a faulty date format. Data imported or pasted into MDU must be of the same format as used when the data is exported. The chapter Section 9.7.1 on page 223 has more information on the format that expected by MDU Tool.
Import failed, Import contents did not match data in current view. See log entry [logentry] for more information.	Error	Importing to MDU.	The content of the file does not match the selection tree MDU was initiated in, or the elements that have been filtered do not match the content of the file.
Save error, Configuration could not be saved, see the log for more information.	Error	Saving in MDU.	The configuration could not be saved, for example because the connection to the server failed. Verify the log.
Found no selection tree elements applicable for update: Possible reason: the selection tree does not contain elements that may be updated with Multi Data Update.	Warning	When trying to find all elements based on tag or path.	If using Find all by path - the selection tree does not contain elements that may be updated with Multi Data Update. If using Find all by tag – no elements have any tag defined OR the selection tree does not contain elements that may be updated with Multi Data Update



Table 98 Error Messages and Warnings in MDU Tool.

Message	Severity	Scenario	Meaning
Found no selection tree elements applicable for update: Possible reason: faulty path format, or no matching elements were found.	Warning	When trying find elements by expression, based on path.	<p>The format is not followed or no elements matching the criteria were found or the tree does not contain elements that may be updated through MDU.</p> <p>The chapter Section 9.7.1 on page 223 has more information on the format that expected by MDU Tool.</p>
Found no selection tree elements applicable for update: Possible reason: faulty tag format, or no matching elements were found.	Warning	When trying find elements by expression, based on tag.	<p>The format is not followed or no elements matching the criteria was found or the tree does not have elements with a defined tag or the tree does not contain elements that may be updated through MDU.</p> <p>The chapter Section 9.7.1 on page 223 has more information on the format that expected by MDU Tool.</p>
Found no selection tree elements applicable for update: Possible reason: selected file has element data that does not match the elements in the selection tree.	Warning	When using find elements by file.	<p>The file has data matching other elements, for example the file is generated from another tree with other tags/paths. Tree configuration may have changed so the tags/paths in the file no longer match. File has been edited manually so the tag/path identifier does not match any longer.</p>
Data pasted with exceptions. All paste contents did not match data in current view. See log entry for more information.	Warning	When pasting into the result table of MDU from system clipboard.	<p>All data in the clipboard did not match the selected view in the result table of MDU. There might have been more, less, faulty or different data on the clipboard than in the result table.</p>
Data imported with exceptions, All import contents did not match data in current view. See log entry [logentry] for more information.	Warning	When importing to MDU.	<p>The file may contain more data than was imported. For example, the filtering may be constrained more specifically than what the file contains.</p>



10 Troubleshooting

Troubleshooting in ERE is carried out by examining the log of events that occur during a session. See Section 10.1 on page 229 for more information on the log function and how it may be configured.

Information presented in Runtime Properties of RMA can also be of use during troubleshooting, see Section 10.2 on page 232.

Some of the most common error messages that occur in RMA are described in Section 10.4 on page 233.

10.1 Event Logging

It is possible to log events in the RMA GUI for troubleshooting purposes. RMA keeps the log as long as the application is running. A persistent log file can be used over several RMA sessions.

The log window is opened by choosing **Util** from the main menu and then choosing **Log**, or by choosing the log icon from the toolbar. The log window is shown Figure 200.

10.1.1 Log Window

An example of the output of the log window is displayed in Figure 200. The log window consists of a list of all log entries in the current session, and an expanded panel describing the currently selected log entry.

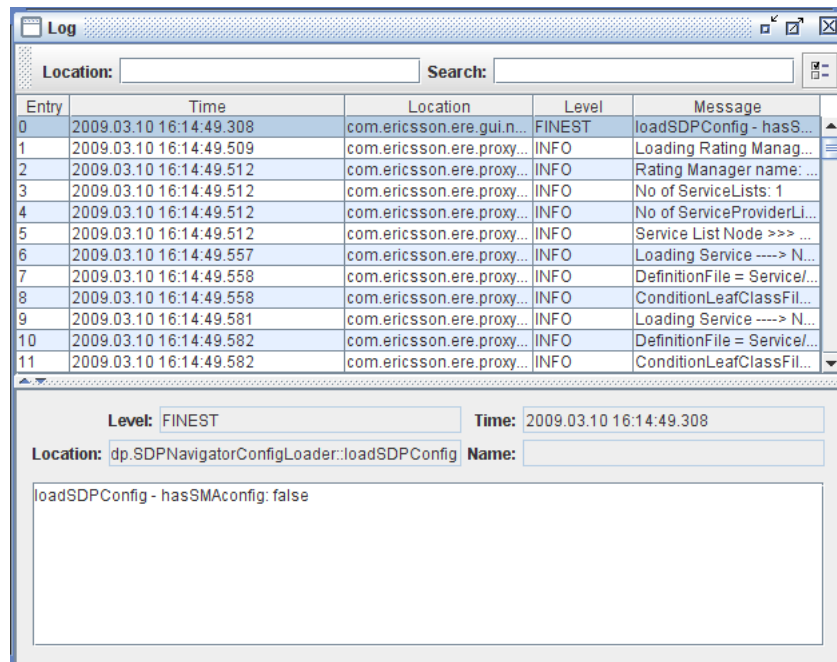


Figure 200 An Example of a Log Window

The log table columns available are described in Table 99:

Table 99 Columns in the Log Table

Options	Description	Format
Entry	The number of the logged event. This is increased by one for each new entry.	Numeric value.
Time	The time specifying the date and time of the logged event.	YYYY-MM-DD hh:mm:ss.SSS.
Location	The place in the code where the event happened.	<package>.<object>::<method>
Level	The severity of the event.	The severity levels are described in Section 10.1.3 on page 232.
Message	Text describing the log entry.	The message header.

There are two filters that make it easier to find events in the log window. Both work on substrings in one of the table columns. The filter **Location:** displays entries pertaining to a certain object or method in the code. The filter **Search:** displays entries containing the entered substring in the entries **Message** field.

All log entries are cleared by right-clicking on any entry and choosing **Clear log window**.



10.1.2 Log Settings

The parameters described here are used to configure the logging function. The settings dialog is opened choosing the **Log Settings** tab in the **RMA Settings** window.

The log settings are described in Table 100.

Table 100 Logger Settings

Options	Description	Format
Max entries	<p>The maximum number of displayed rows in the log window.</p> <p>By default, the maximum number of rows is 300, and the minimum number is 1. The log will only contain the entries generated since the RMA has been started.</p>	<p>Numeric field.</p> <p>Default value: 300.</p>
Minimum log level	<p>The minimum level of the events shown in the log window. Events more severe than this will be displayed, events less severe will be hidden.</p> <p>The options available are those described in Section 10.1.3 on page 232.</p>	
Use Persistent Log File	Check box indicating if the log should be written to file that will be retained over several RMA sessions.	<p>Check box checked - use persistent log file.</p> <p>Check box cleared - do not use persistent log file.</p>
File name	<p>The file name of the log file. This field is only applied if the Use Persistent Log File check box is selected.</p> <p>To choose an existing file select the file browser button to the right of the field and browse to the file in question.</p>	
Output	Option buttons indicating the format of the log file. This field is only applied if the Use Persistent Log File check box is checked.	<p>The output formats available are:</p> <ul style="list-style-type: none"> • XML • Plain



10.1.3 Severity Levels

Each log entry has a severity level assigned to it. This level is assigned in the application that integrates ERE.

Set the lowest severity level of events to be logged in RMA by right-clicking on the list of log entries and choosing a severity level from the menu displayed. All events with the same or a higher severity level to this will be logged.

The following log levels are available, ordered by amount of details provided.

- Finest
- Finer
- Fine
- Config
- Info
- Warning
- Severe

Setting the severity level to **Off** disables the logging function.

10.2 Runtime Properties

The runtime properties show the system information for the machine that RMA is currently running on. This information can be helpful when troubleshooting RMA.

1. Choose **Help** and **About** from the main menu.
2. Choose the **Runtime Properties** tab.
3. Choose either **Print...** to print the information on paper or **Save to file...** to save the information to file.

The runtime properties tab also shows information on how much memory is allocated to the RMA on the server, and how much of this memory currently used by the application. See Section 10.3 on page 232 for more information on memory use.

10.3 Memory Use

The amount of memory on the server allocated to RMA is visible in the Runtime Properties tab, see Section 10.2 on page 232. The amount of memory used by the application is also visible, both as in kB and as a percentage of the total available memory.



If the Show Memory Usage option button is selected, see **RMA Settings > General** in Section 4.2 on page 19, the percentage of allocated memory currently in use is shown graphically in the information bar.

It is recommended to keep the amount of memory less than 75% of the maximum value. Otherwise there may be problems opening or saving larger selection trees.

The memory use may be kept low by closing selection trees and services that are not currently being worked on.

10.4 Common Error Messages

The following are some of the most common error messages returned by the RMA.

10.4.1 Internal ERE Could Not Start, Address Already in JVM Bind

When multiple instances of RMA are started on the same workstation, it is not possible to work against the Internal ERE any other instance than the first RMA GUI that was started since the port for the Internal ERE is busy.

This results in an error message on start-up. RMA will launch as usual and it will be possible to connect to other ERE instances than the Internal ERE and work with them as usual.



Figure 201 Internal ERE Could Not Be Started





11 Online Help

Each input field in the RMA GUI consists of parameter data. The information is also available by pressing the **F1** key while the cursor is placed in the input field.

The complete online help can also be accessed by choosing the menu option **Help** and then **RMA Help**. The online help contains the same information as this document.

11.1 Online Help Navigation

The online help navigation overview is shown and described below, see Figure 202 and Table 101.



Figure 202 Icons in the Online Help Navigation

Table 101 Online Help Navigation Overview Explained

Number in picture	Icon name	Description
1	Previous	Displays the previous help section
2	Next	Displays the next available help section
3	Print	Prints the displayed help section using the default printer settings
4	Page Setup	Opens the page setup window
5	Table of Contents	Displays the content of the online help
6	Index	Displays the index file of the online help
7	Search	Possibility to search the online help

11.2 Rating Manager Connection Input Fields

To get more information about the parameters relating to the connection of a Rating Manager in the ERE Navigator, see Section 4.5 on page 23.



11.3 Rating Manager Input Field

To get more information about the parameters relating to the Rating Manager, see Section 4.5.2 on page 24.



12 Third Party License Agreements

ERE uses some freeware third-party product software. This chapter contains the license agreements for these products.

12.1 Apache

Code Generation Library (CGLIB) is licensed under the Apache 2.0 license agreement.

12.1.1 Copyright Notice

Copyright 2007-2010 Ericsson AB

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

12.1.2 License

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by



contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise



transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - You must cause any modified files to carry prominent notices stating that You changed the files; and
 - You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
 - If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

END OF TERMS AND CONDITIONS

12.2 ASM

The ASM library is a project of the ObjectWeb consortium.

12.2.1 Copyright Notice

Copyright 2000-2005 INRIA, France Telecom. All rights reserved.

12.2.2 License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:



1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.





Glossary

ASCII

American Standard Code for Information Interchange

BCD

Binary Coded Decimal

CDR

Charging Data Record

CGLIB

Code Generation Library

CIDR

Classless Inter-Domain Routing

ERE

Ericsson Rule Engine - a framework for building a rule engine

Field

Defines an identifier and the form of a piece of data. Configured in the service

GUI

Graphical User Interface

Internal ERE

An ERE server instance that is running on the local machine, used for testing purpose

IP

Internet Protocol

JAR

Java ARchive

JDBC

Java Database Connectivity

MDU

Multi Data Update, an ERE tool that allows data in selection trees to be updated easily.

MMS

Multimedia Messaging Service

Modifier

A element of the selection tree in which data is evaluated or modified.

MSISDN

Mobile Station Integrated Services Network Number

Node

A container of other selection tree elements that can add on behavior for the contained elements

Plugin

See selection tree element

Rating Period

Specifies a start time and name for one selection tree

Rating Plan

Rating period container with a specified service

RMA

Rating Management Application - The GUI used for configuring ERE

SDD Tool

Selection Tree Difference Detection Tool

SDF

Service Defined Field

Selection Tree

Configuration rules, built of selection tree elements

Selection tree element

A building block, for example a condition or modifier. Defined in the service and used in the selection tree

Service

Contains definitions of data and selection tree elements

**Service Provider**

Rating plan container

SMS

Short Message Service

Structure

Forms the root of a selection tree and can add on behavior for the whole tree

TDF

Tree Defined Field - a definition of data, where the definition is done in a selection tree.

XML

Extensible Markup Language



Reference List

Ericsson Documents

- [1] SDP Customer Product Information Overview, 1/1551-EN/LZN 741 0126 Uen