EMC® Smarts® Notification Module

Version 9.2

User Guide

P/N 300-999-856
REV 01

EMC²

Copyright © 1996 - 2013 EMC Corporation. All rights reserved. Published in the USA.

Published March, 2013

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC^2, EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (https://support.emc.com).

CONTENTS

Preface

Chapter 1 Introduction

EMC Smarts Notification Module overview 14
EMC Smarts Service Assurance Manager Adapter Platform architecture 15
EMC Smarts NOTIF integration with Adapter Platform architecture 16
EMC Smarts NOTIF architecture 17
Advantages of EMC Smarts NOTIF architecture 18
EMC Smarts NOTIF processor functionality 19
EMC Smarts NOTIF Editor functionality 19
Local *.ncf file configuration options (File menu) 20
Connecting to a remote server through the EMC Smarts NOTIF Editor (Remote › Manage connections) 20
Remote *.ncf file configuration options (Remote menu) 22

Chapter 2 Event Configuration Information Processing

ECI processing overview 24
Raw event normalization 26
Pushing events into EMC Smarts NOTIF 27
ECI event matching 28
Designing ECI naming conventions 29
ECI notification naming 30
Enumeration of keyword/value pairs 31
Value substitutions 32
Setting up ECI configuration parameters 32
Event Options values 36
Copying ECI configuration parameters 43

Chapter 3 Notification Configuration Information Processing

NCI processing overview 46
NCI notification matching 47
Designing NCI naming conventions 50
Setting up NCI configuration parameters 51
Mixing aggregation and causes 61
Copying NCI configuration parameters 62

Chapter 4 Using EMC Smarts NOTIF

Starting EMC Smarts NOTIF 66
Using the EMC Smarts NOTIF Editor 70
EMC Smarts NOTIF Editor tree view 70
Main menu options 73
Generating EMC Smarts NOTIF documentation 95
Disabling EMC Smarts NOTIF 96

Appendix A **Substitution Scripting and Values**

Script parsing overview for substitution features 98
Built-in operators 99
Built-in functions 99
Substitution values for ECI configuration parameters..... 100

Appendix B **ASL and Java Interfacing with EMC Smarts NOTIF**

Writing ASL for use with EMC Smarts NOTIF ECI processing 104
Writing ASL for use with EMC Smarts NOTIF NCI processing..... 106
Writing a Java class for use with EMC Smarts NOTIF event processing 107
EMC Smarts NOTIF Java API..... 108
An example plug-in processor 109
Additional notes..... 109

Index

FIGURES

	Title	Page
1	Standard EMC Smarts Service Assurance Manager Adapter Platform architecture	15
2	Overlay of EMC Smarts NOTIF functions on Adapter Platform architecture	17
3	EMC Smarts NOTIF ECI/NCI structure in an EMC Smarts environment	18
4	EMC Smarts NOTIF Editor functionality	21
5	ECI processing and NCI processing steps within EMC Smarts NOTIF	24
6	General tab of the ECI Setup dialog box	32
7	Example of mouse-hover tool in the EMC Smarts NOTIF Editor	35
8	Event Setup tab of the ECI Setup dialog box	35
9	Event Options tab of the ECI Setup dialog box	36
10	User Defined tab of the ECI Setup dialog box	38
11	Calc'd Values tab of the ECI Setup dialog box	38
12	Edit Calculated Value dialog box	40
13	Example of adding a Calculated Value to an ECI	41
14	Example of editing a Calculated Value for an ECI	41
15	ECI copy feature	43
16	NCI processing steps	46
17	NCI notification matching process	49
18	General tab of the NCI Setup dialog box	52
19	ClearsForOthers tab in the NCI Setup dialog box	53
20	ClearedByOthers tab in the NCI Setup dialog box	53
21	Add Causes Relationship dialog box	54
22	Causality tab shows NCI-Router.Alarm-145-3-116 cause NCI-Router.Alarm-7-2-5	55
23	Causality tab shows NCI-Router.Alarm-7-2-5 cause NCI-Router.Alarm-145-3-116	55
24	NCI causality relationship occurs on an instance related to a notified instance	57
25	NCI causality relationship occurs on an element related to a notified element	59
26	AggregatesFrom tab of the NCI Setup dialog box	60
27	AggregatesTo tab of the NCI Setup dialog box	61
28	NCI copy feature	63
29	Connect to a Server dialog box	68
30	Server connection Password dialog box	68
31	Remote Server Settings dialog box	69
32	EMC Smarts NOTIF configuration objects in the EMC Smarts NOTIF Editor topology tree	70
33	Enumerated value mapping for EVM-.1.3.6.1.4.1.9.9.385.1.3.1	71
34	SNMP trap configuration parameters for the ECI named 1.2.3.4.5.6.47.108.10_6_10..	72
35	Configuration parameters for the NCI named Router.Alarm-4-84-70	72
36	Settings for the NotifPackage named SNMP_Traps	73
37	New Configuration window in the EMC Smarts NOTIF Editor	74
38	Open Notif Config File dialog box	74
39	Merge Notif Config file dialog box	75
40	Merge Conflict dialog box	75
41	Save dialog box	76
42	Loading trap_mgr.conf dialog box	77
43	Create EventConfigInfo feature	78
44	Create NotifConfigInfo feature	79
45	Create EnumValueMapping feature	79
46	Create Package feature	79
47	Directory dialog box	81
48	Select Remote File	81
49	Push Complete dialog box	82
50	Reload Complete dialog box	82

51 Select Remote File dialog box 83
52 Push complete for package dialog box 83
53 Loading a running configuration from a remote server 84
54 Running configuration from remote server loaded into EMC Smarts NOTIF Editor 85
55 File dialog box 85
56 Loading an .ncf file from a remote server 86
57 .ncf file from a remote server is now loaded into the EMC Smarts NOTIF Editor 86
58 Remote Server Settings dialog box 87
59 Blank Manage Connections dialog box 89
60 Input dialog box for remote server connection reference name 89
61 Manage Connections dialog box with remote server connection reference name 89
62 Selecting a domain in the Manage Connections dialog box 90
63 Connection confirmation dialog box 90
64 Completed connection setup data in the Manage Connections dialog box 91
65 Loading Instances from an Import File dialog box 92
66 Overwrite Configuration dialog box 93
67 Overwrite changed configuration dialog box 93
68 EMC Smarts NOTIF HTML documentation 95

PREFACE

As part of an effort to improve its product lines, EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your EMC technical support professional if a product does not function properly or does not function as described in this document.

NOTICE

This document was accurate at publication time. Go to EMC Online Support (https://support.emc.com) to ensure that you are using the latest version of this document.

Purpose

This guide provides instructions for using the EMC Smarts Notification Module (EMC Smarts NOTIF) application within an EMC Smarts environment to process external events. EMC Smarts NOTIF is a configuration tool and run-time product for raw event and notification processing. Events and notifications are configured through the flexible graphical user interface, the EMC Smarts Notification Module Configuration Editor (EMC Smarts NOTIF Editor).

In addition to the configuration guides for specific components, administrators should also read the EMC Smarts Service Assurance Manager Deployment Guide and the *EMC Smarts System Administration Guide*.

Audience

This guide is intended for users of EMC Smarts products with domain knowledge of how an event is to be processed and filtered. The person deploying and configuring EMC Smarts NOTIF should be familiar with the EMC Smarts Service Assurance Manager (Global Manager or SAM), Adapter Platform, and notifications. This guide is intended for any user who needs to manage and import external events into EMC Smarts Service Assurance Manager products.

EMC Smarts Service Assurance Manager installation directory

In this document, the term BASEDIR represents the location where EMC Smarts software is installed:

- For UNIX, this location is: /opt/InCharge/*‹product›*.
- For Windows, this location is: C:\InCharge*‹product›*.

The *‹product›* represents the EMC Smarts product to which the product belongs. For example, on UNIX operating systems, EMC Smarts Service Assurance Manager is, by default, installed to /opt/InCharge/SAM/smarts. On Windows operating systems, this product is, by default, installed to: C:\InCharge\SAM\smarts. This location is referred to as BASEDIR/smarts.

Optionally, you can specify the root of BASEDIR to be something other than /opt/InCharge (on UNIX) or C:\InCharge (on Windows), but you cannot change the *‹product›* location under the root directory.

The *EMC Smarts System Administration Guide* provides more information about the directory structure of EMC Smarts software.

EMC Smarts Service Assurance Manager

The EMC Smarts Service Assurance Manager includes the following products:

- Service Assurance Manager (Global Manager), includes Business Impact Manager (BIM) and Failover System
- Global Console
- Business Dashboard
- Companion UI
- SAM Native Adapters:
  - Service Assurance Manager Notification Adapters (E-Mail Notifier Adapter, Script Notifier Adapter, SNMP Trap Notifier Adapter, Log File Notifier Adapter)
  - Adapter Platform (Adapter Platform server, Syslog Adapter, SNMP Trap Adapter, sm_ems command-line interface)
  - XML Adapter
  - EMC Smarts Adapter for UIM/O

Related documentation

In addition to this document, EMC Corporation provides a help system for command line programs as well as product documentation.

Help for command line programs

Descriptions of command line programs are available as HTML pages. The index.html file, which provides an index to the various commands, is located in the BASEDIR/smarts/doc/html/usage directory.

EMC Smarts documentation

Readers of this guide may find the following related documentation helpful:

- *Smarts Foundation EMC Data Access API (EDAA) Programmer Guide*
- *EMC Smarts Common Information Model (ICIM) 1.11 Reference for Non-Service Assurance Manager Products*
- *EMC Smarts Common Information Model (ICIM) Reference for Service Assurance Manager*
- *EMC Smarts Foundation Documentation Portfolio*
- *EMC Smarts System Administration Guide*
- *EMC Smarts ASL Reference Guide*
- *EMC Smarts Perl Reference Guide*

- *EMC Smarts Dynamic Modeling Tutorial*
- *EMC Smarts MODEL Reference Guide*
- *FlexNet Publisher License Administration Guide*
- *Smarts Foundation EMC Data Access API (EDAA) Programmer Guide*
- *EMC Smarts Common Information Model (ICIM) 1.11 Reference for Non-Service Assurance Manager Products*
- *EMC Smarts Common Information Model (ICIM) Reference for Service Assurance Manager*
- *EMC Smarts Foundation Documentation Portfolio*
- *EMC Smarts System Administration Guide*
- *EMC Smarts ASL Reference Guide*
- *EMC Smarts Perl Reference Guide*
- *EMC Smarts Dynamic Modeling Tutorial*
- *EMC Smarts MODEL Reference Guide*

NOTICE

*FlexNet Publisher License Administration Guide*These documents are updated periodically. Electronic versions of the updated manuals are available on the EMC online support website:
https://support.emc.com

EMC Smarts Service Assurance Manager documentation

The following documents are relevant to users of the EMC Smarts Service Assurance Manager:

- *EMC Smarts Service Assurance Manager Release Notes*
- *EMC Smarts Open Source License and Copyright Information*
- *EMC Smarts Service Assurance Manager Introduction*
- *EMC Smarts Installation Guide for SAM, IP, ESM, MPLS, VoIP, and NPM Managers*
- *EMC Smarts Service Assurance Manager Deployment Guide*
- *EMC Smarts Service Assurance Manager Configuration Guide*
- *EMC Smarts Service Assurance Manager Operator Guide*
- *EMC Smarts Service Assurance Manager Dashboard Configuration Guide*
- *EMC Smarts Business Impact Manager User Guide*
- *EMC Smarts Failover System User Guide*
- *EMC Smarts Service Assurance Manager Notification Adapters User Guide*
- *EMC Smarts Service Assurance Manager Adapter Platform User Guide*
- *EMC Smarts Adapter for Unified Infrastructure Manager/Operations Configuration Guide*

- *EMC Smarts XML Adapter User Guide*
- *EMC Smarts Notification Module User Guide*
- *EMC Smarts Service Assurance Manager Troubleshooting Guide*
- *EMC Smarts Companion UI Online Help*
- *EMC Smarts Companion UI Open Source License and Copyright Information for GPLv3 as Included with a Distribution of SLES 11*
- *EMC Smarts Service Assurance Manager Documentation Portfolio*

The *EMC Smarts Documentation Catalog* provides documentation resources for other EMC Smarts products.

Conventions used in this document

EMC uses the following conventions for special notices:

NOTICE

NOTICE is used to address practices not related to personal injury.

Typographical conventions

EMC uses the following type style conventions in this document:

Bold	Use for names of interface elements, such as names of windows, dialog boxes, buttons, fields, tab names, key names, and menu paths (what the user specifically selects or clicks)
Italic	Use for full titles of publications referenced in text
`Monospace`	Use for: • System output, such as an error message or script • System code • Pathnames, filenames, prompts, and syntax • Commands and options
`Monospace italic`	Use for variables.
`Monospace bold`	Use for user input.
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections — the bar means “or”
{ }	Braces enclose content that the user must specify, such as x or y or z
...	Ellipses indicate nonessential information omitted from the example

Where to get help

EMC support, product, and licensing information can be obtained as follows:

Product information — For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

https://support.emc.com

Technical support — Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to:

`techpubcomments@emc.com`

CHAPTER 1
Introduction

This chapter consists of the following topics:

- EMC Smarts Notification Module overview 14
- EMC Smarts Service Assurance Manager Adapter Platform architecture 15
- EMC Smarts NOTIF integration with Adapter Platform architecture 16
- EMC Smarts NOTIF architecture 17
- EMC Smarts NOTIF processor functionality..... 19
- EMC Smarts NOTIF Editor functionality..... 19

EMC Smarts Notification Module overview

The EMC® Smarts® Notification Module (EMC Smarts NOTIF) augments the EMC Smarts solutions with event management features that are configurable through a graphical user interface. The EMC Smarts NOTIF user interface is referred to as the EMC Smarts Notification Module Configuration Editor (EMC Smarts NOTIF Editor). EMC Smarts NOTIF replaces the internal event and notification processing features of the standard EMC Smarts Service Assurance Manager Adapter Platform. EMC Smarts NOTIF provides the user with event and notification configuration access, maintenance, and documentation capabilities without requiring a complete understanding of the underlying products, and without requiring extensive application-specific coding.

EMC Smarts NOTIF represents an improvement over the standard EMC Smarts trap processing design. To configure the standard SNMP Trap Adapter to process traps, you must decide which traps should become notifications and how the notifications should appear at the Global Console. You can do this by editing the trap_mgr.conf file and then restarting the server. EMC Smarts NOTIF makes this process much more user-friendly and robust. With EMC Smarts NOTIF, you can make instantaneous trap processing configuration changes through the EMC Smarts NOTIF Editor without restarting any servers.

Efficient event and notification processing presents the most important information to the operator by using an EMC Smarts Service Assurance Global Console. EMC Smarts NOTIF enables the user to easily optimize the flow of events and notifications sent through any EMC Smarts system. Advantages of using EMC Smarts NOTIF include the following:

- Handles notifications in a consistent and extensive manner.
- Facilitates configuration of events and notifications through a flexible, user-friendly interface, the EMC Smarts NOTIF Editor.
- Allows for real-time reconfiguration of event and notification processing without server restarts.
- Applies to any event source.
- Integrates with standard EMC Smarts Service Assurance Manager (SAM) and Adapter Platform (OI) repositories and servers.
- Supports current EMC Smarts Service Assurance Manager Adapter Platform standard SNMP event functions (that is, expiration clearing, unknown agents, logging specifications, notification field setting).
- Provides converter tools for standard configuration sources (for example, trap_mgr.conf file).
- Packages event and notification configurations. This is useful for modular roll-out of configurations.
- Automatically generates an HTML-based document that provides information on all internal event and notification configuration objects for review and documentation purposes.
- Requires no knowledge of the EMC Smarts Adapter Scripting Language (ASL).

EMC Smarts Service Assurance Manager Adapter Platform architecture

To understand the event management capabilities and advantages that EMC Smarts NOTIF provides in an EMC Smarts implementation, it is important to understand the EMC Smarts Service Assurance Manager Adapter Platform architecture and the standard EMC Smarts trap processing design.

Figure 1 on page 15 is an example of a basic SAM configuration. The Adapter Platform imports topology and event information from third-party applications and prepares it for use by the Service Assurance Manager. Data from third-party applications should not be sent directly to SAM. The Adapter Platform creates notifications from the incoming information and can even associate these notifications to topology information supplied by another EMC Smarts application, such as the EMC Smarts IP Availability Manager as shown in Figure 1 on page 15.

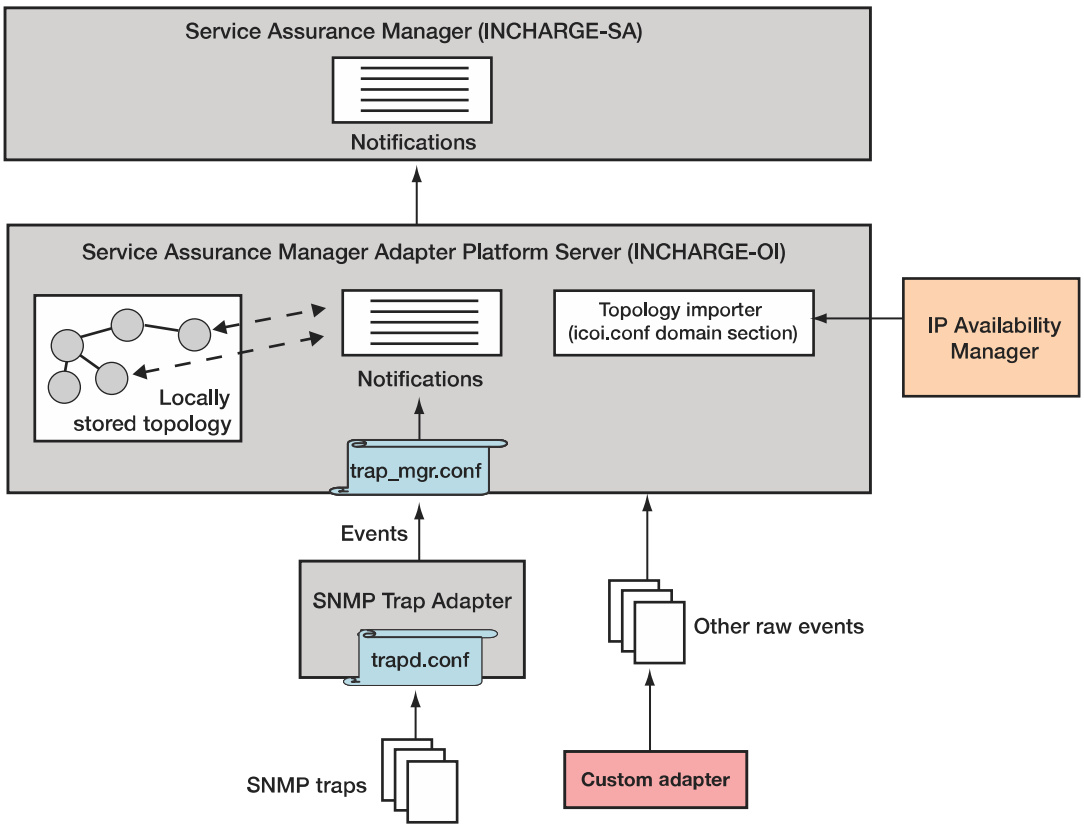


Figure 1 Standard EMC Smarts Service Assurance Manager Adapter Platform architecture

The Adapter Platform consists of the SNMP Trap Adapter, as shown in Figure 1 on page 15. The SNMP Trap Adapter parses SNMP traps received and generates notifications based on the contents of the traps. The SNMP Trap Adapter can receive SNMP v1, v2c, or v3 traps. However, it forwards only v1 or v2c traps. If SNMPv3 traps are received, they are converted to one of the other formats before being sent to a domain manager or to another trap receiver.

NOTICE

The *EMC Smarts Service Assurance Manager Adapter Platform User Guide* provides details on the Adapter Platform.

The SNMP Trap Adapter can be configured to handle many different types of traps through two configuration files associated with the SNMP Trap Adapter:

- trapd.conf — This is the configuration file for the EMC Smarts SNMP Trap Adapter. This conf file determines what port number the trap adapter listens to for traps, and enables trap forwarding to EMC Smarts domain managers (like the IP Availability Manager) and the Adapter Platform.
- trap_mgr.conf — This configuration file determines how specific traps are mapped into notifications that are sent to the Global Console.

EMC Smarts NOTIF integration with Adapter Platform architecture

EMC Smarts NOTIF allows you to filter and forward events and perform other event management configuration through Event Configuration Information (ECI) objects. The event configuration features of EMC Smarts NOTIF replace the trap_mgr.conf file in the Adapter Platform (OI) server shown in Figure 2 on page 17, which performs the event filtering and determines which traps are forwarded for further processing. The EMC Smarts NOTIF functions are overlaid on the standard EMC Smarts Service Assurance Manager Adapter Platform architecture as shown in Figure 2 on page 17. EMC Smarts NOTIF can be installed on an Adapter Platform server or a SAM server.

EMC Smarts NOTIF also allows you to process notifications through Notification Configuration Information (NCI) objects. The notification processing features of EMC Smarts NOTIF replace the internal notification processing in the Adapter Platform or SAM server, depending on where EMC Smarts NOTIF is installed, as shown in Figure 2 on page 17.

NOTICE

"EMC Smarts NOTIF architecture" on page 17 provides details on ECI and NCI objects.

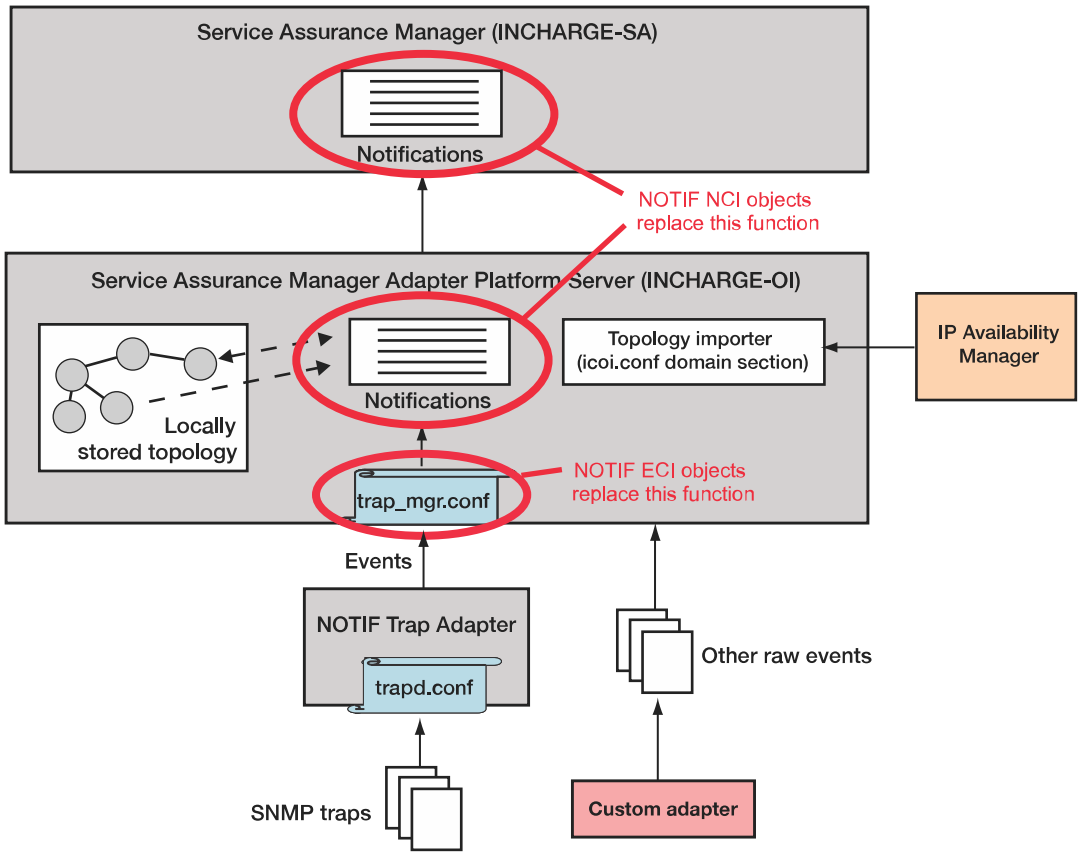


Figure 2 Overlay of EMC Smarts NOTIF functions on Adapter Platform architecture

EMC Smarts NOTIF architecture

There are two main components of EMC Smarts NOTIF. The first is the real-time component which processes SNMP traps or other raw event types according to configuration settings. This configuration processing is performed on raw events by the Event Configuration Information (EventConfigInfo or ECI) objects. ECI objects determine whether a particular event will be processed, that is, whether a notification for an event will be created. ECI processing is discussed in Chapter 2, "Event Configuration Information Processing."

The second EMC Smarts NOTIF component is notification processing, which is configured by the Notification Configuration Information (NCI) objects. NCI objects determine how notifications are processed. NCI processing is discussed in Chapter 3, "Notification Configuration Information Processing."

It is important to note the separation of raw event processing (through ECI objects) versus notification processing (through NCI objects) since this allows EMC Smarts NOTIF to operate in various architectures. The EMC Smarts NOTIF Editor allows the user to manipulate the configuration settings for both ECI and NCI objects.

In an EMC Smarts implementation, EMC Smarts NOTIF can reside on a SAM server or an Adapter Platform Open Interface (OI) server. Figure 3 on page 18 shows how ECI and NCI objects are used in an EMC Smarts NOTIF solution where low-level raw events are filtered

and processed, and notifications are grouped within the Adapter Platform. An additional EMC Smarts NOTIF process is run within the SAM server to combine and group notifications from the adapter servers and other (analysis) servers.

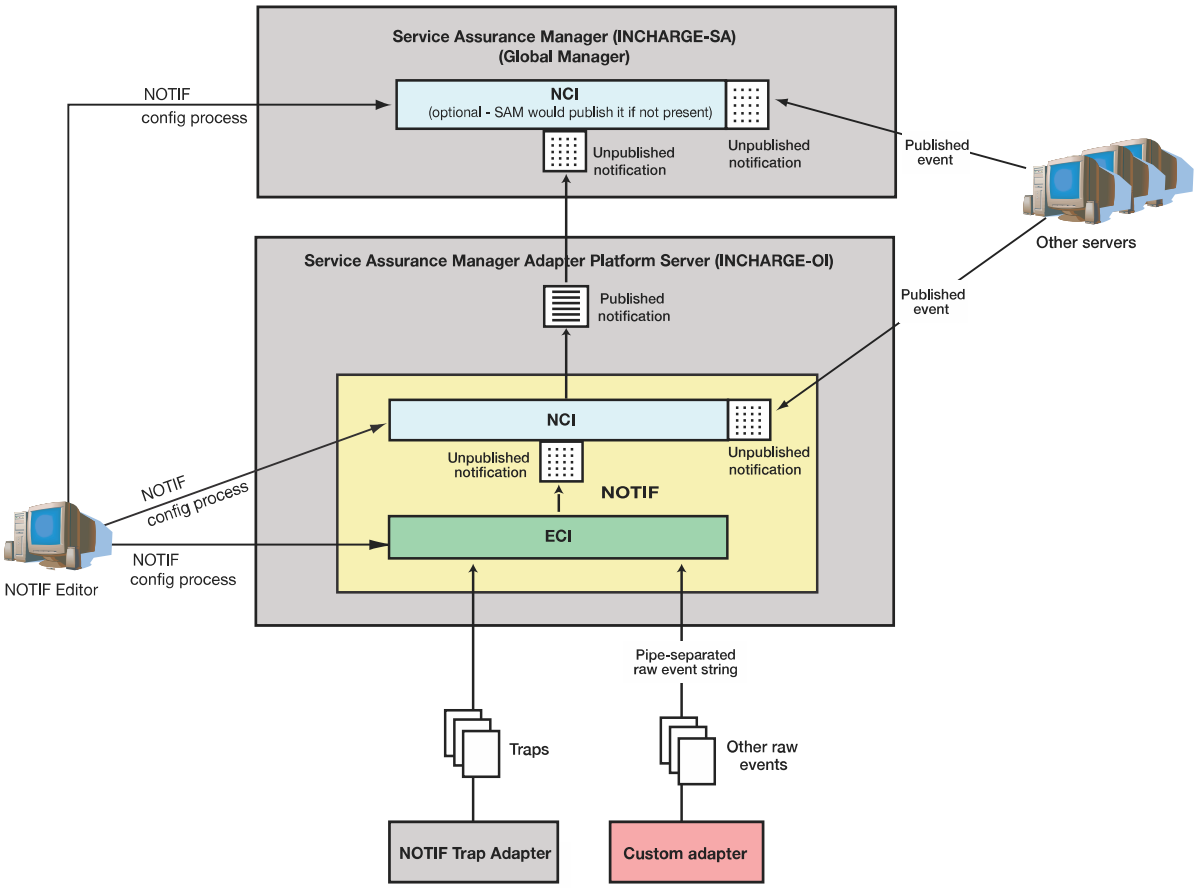


Figure 3 EMC Smarts NOTIF ECI/NCI structure in an EMC Smarts environment

Advantages of EMC Smarts NOTIF architecture

In the EMC Smarts NOTIF Editor, you configure ECI objects to determine how you want to handle particular events. In the ECI setup window, in the EMC Smarts NOTIF Editor, you can select whether a particular raw event should be handled. If the event is to be processed, select the Active checkbox in the EMC Smarts NOTIF Editor as shown in Figure 6 on page 32. When you select whether a particular event is Active, you save the configuration to the local Adapter Platform server (File › Save Configuration or File › Save Configuration As... in the EMC Smarts NOTIF Editor), and you do not have to restart the Adapter Platform server.

The EMC Smarts NOTIF Editor also allows you to make changes to varbinds associated with ECI configuration parameters through the ECI setup window in the EMC Smarts NOTIF Editor. Without the EMC Smarts NOTIF Editor, you would have to do this by writing hook scripts.

"Setting up ECI configuration parameters" on page 32 provides details on how to configure ECI objects to handle incoming events.

EMC Smarts NOTIF processor functionality

When the EMC Smarts NOTIF processor starts up on a server or the Remote › Reload server option in the EMC Smarts NOTIF Editor is selected, EMC Smarts NOTIF browses the *BASEDIR*/smarts/conf/‹*config*› and local/conf/‹*config*› directories for files that end in ".ncf". The *.ncf files are EMC Smarts NOTIF configuration files. EMC Smarts NOTIF loads each*.ncf file into the EMC Smarts NOTIF processor and combines them to create the running configuration on the Adapter Platform or SAM server where EMC Smarts NOTIF is running.

NOTICE

The use of *.ncf files in EMC Smarts NOTIF is a major architectural change from previous versions of NOTIF. EMC Smarts NOTIF now loads configurations from *.ncf files, one at a time, rather than connect to an EMC Smarts repository.

EMC Smarts NOTIF configuration files (*.ncf files) are XML files that you can edit using the EMC Smarts NOTIF Editor. "EMC Smarts NOTIF Editor functionality" on page 19 provides an overview of how the EMC Smarts NOTIF Editor works.

When EMC Smarts NOTIF reads an *.ncf configuration file and loads the entire configuration from the configuration directories (that is, the /conf/‹*icoi or /ics*› or /local/conf/‹*icoi or /ics*› directories), the cumulative configuration is written to an *.ncf file in the local/logs directory (for example, the INCHARGE-OI.‹*configfilename*›.ncf file). In this way, you can determine the specific EMC Smarts NOTIF configuration elements that are running on a particular server. The *.ncf file written to the local/logs directory is generated when you either start up EMC Smarts NOTIF or select the Remote › Reload server option in the EMC Smarts NOTIF Editor.

NOTICE

When you install the EMC Smarts Service Assurance Manager version 9.2, EMC Smarts NOTIF is not enabled by default. You must start EMC Smarts NOTIF by performing the steps described in "Starting EMC Smarts NOTIF" on page 66.

EMC Smarts NOTIF Editor functionality

The EMC Smarts NOTIF Editor is designed to allow you to manipulate EMC Smarts NOTIF configuration files (*.ncf), both on the local machine that is running the EMC Smarts NOTIF Editor as well as remote servers. The EMC Smarts NOTIF Editor accesses local configuration *.ncf files under the File menu option, while remote operations are located under the Remote menu option. "Main menu options" on page 73 provides details on The EMC Smarts NOTIF Editor menu options.

The EMC Smarts NOTIF Editor provides key functions through the main menu such as retrieving configurations from remote servers, pushing (or storing) configurations to remote servers, and managing connections to remote servers. Figure 4 on page 21 provides a schematic overview of the EMC Smarts NOTIF Editor functionality.

Local *.ncf file configuration options (File menu)

On the local server, you can edit an. *ncf file through the following File menu options:

- New Configuration
- Open Configuration
- Merge Configuration
- Save Configuration
- Save Configuration As
- Close Configuration

You can also import trap_mgr.conf files to your local server through the Import trap_mgr.conf File menu option. This option allows you to load the trap_mgr.conf files used to configure a TRAP-INCHARGE-OI server (SNMP Trap Adapter) into EMC Smarts NOTIF to replicate the standard file-based functionality.

NOTICE

"File menu options" on page 73 provides more information on File menu options.

Connecting to a remote server through the EMC Smarts NOTIF Editor (Remote › Manage connections)

Connections are defined within the EMC Smarts NOTIF Editor to access remote servers. Multiple connections may be defined and given a reference name for convenience. The Remote › Manage connections menu option is used to add remote server connections in the EMC Smarts NOTIF Editor. When you select this option, you type a connection reference name and connection details such as broker, domain, and user names.

NOTICE

"Manage connections..." on page 88 provides more information on how to set up remote server connections.

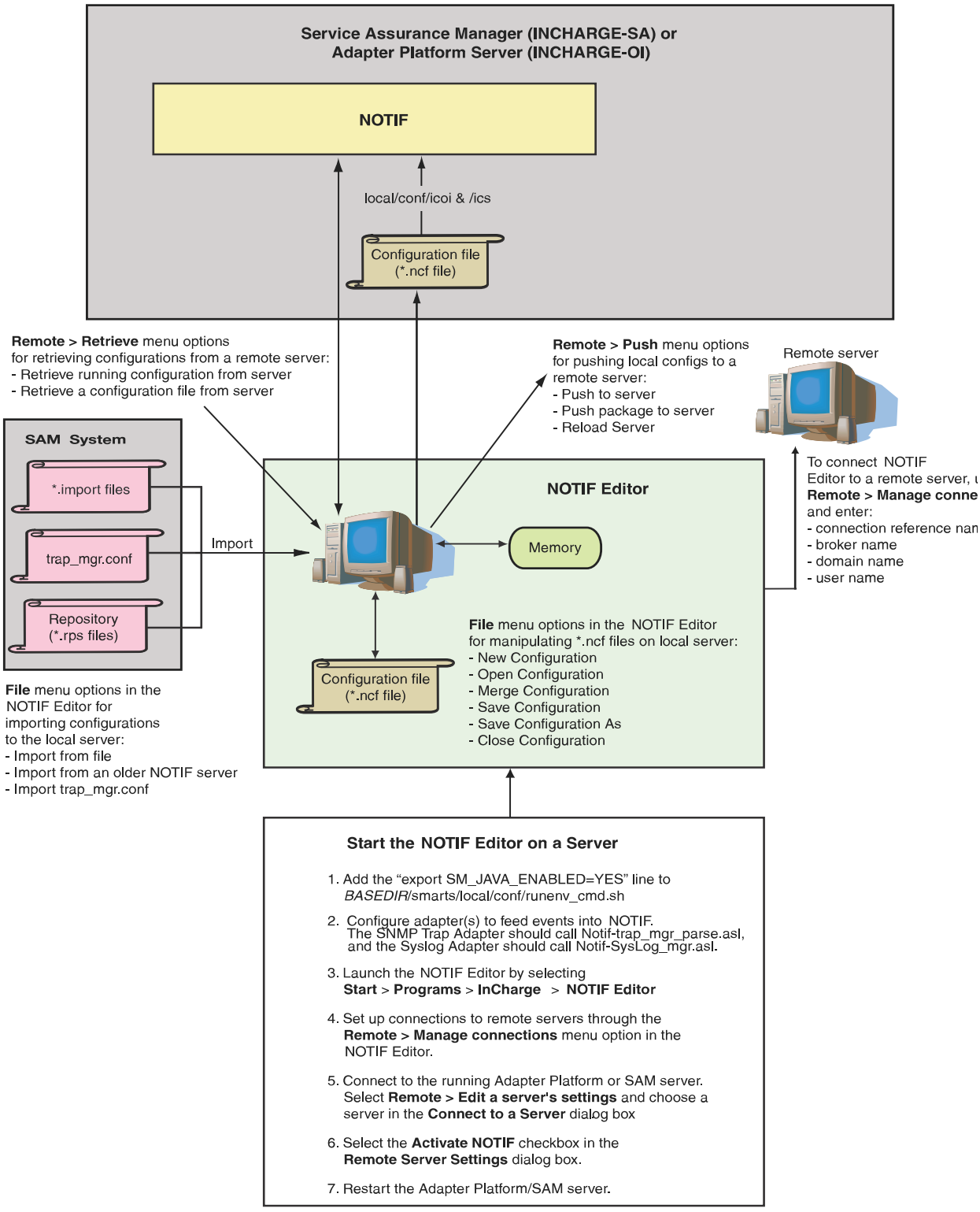


Figure 4 EMC Smarts NOTIF Editor functionality

Remote *.ncf file configuration options (Remote menu)

After you have established the remote server connection through the Remote › Manage connections option, push or store a local *.ncf configuration file or package to a remote server through the following EMC Smarts NOTIF Editor options:

- Push to server
- Push package to server

After the configurations are pushed to a remote server, they do not become active on the remote server until you select the Remote › Reload Server option.

The Remote menu options associated with retrieving a running configuration or a specific *.ncf file from a remote server to a local server are:

- Retrieve running configuration from server
- Retrieve a configuration file from server

This Remote › Edit a server's settings option is used to change server settings on remote servers. One of the settings you can change on a remote server is whether EMC Smarts NOTIF is active on that server. "Starting EMC Smarts NOTIF" on page 66 provides details on the process for activating EMC Smarts NOTIF on a remote server by using the Edit a server's settings option.

NOTICE

"Remote menu options" on page 80 provides more information on Remote menu options.

CHAPTER 2
Event Configuration Information Processing

This chapter describes Event Configuration Information (EventConfigInfo or ECI) processing.

This chapter consists of the following topics:

- ECI processing overview 24
- Raw event normalization 26
- Pushing events into EMC Smarts NOTIF 27
- ECI event matching 28
- Designing ECI naming conventions 29
- Enumeration of keyword/value pairs 31
- Value substitutions 32
- Setting up ECI configuration parameters 32
- Copying ECI configuration parameters 43

ECI processing overview

Event Configuration Information (ECI) processing involves two steps:

- Determining whether a raw event should be processed to create an unpublished notification that is passed to notification processing (that is, filtering events at the lowest level possible).
- Determining the instance/class/event name of the notification that should be created if the event is processed.

Figure 5 on page 24 shows the overall flow of ECI processing steps. Chapter 3, "Notification Configuration Information Processing." provides details on how ECI processing steps relate to the NCI processing steps.

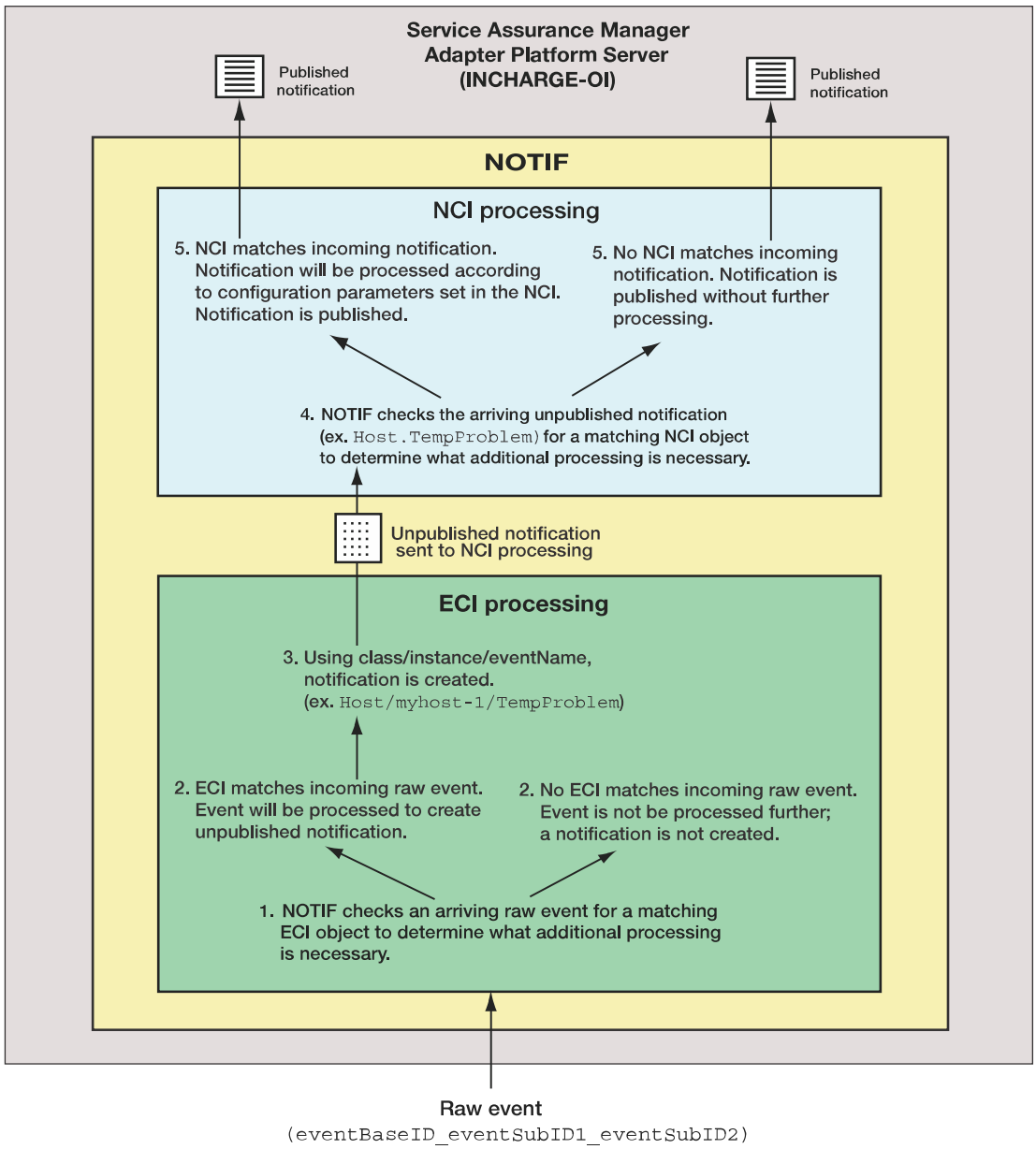


Figure 5 ECI processing and NCI processing steps within EMC Smarts NOTIF

Event information is passed to EMC Smarts NOTIF through a common structure. This packaging allows for common information to logically identify an event and pass additional information that may be used to further convert raw information into a specific notification.

The packaging of the event data is performed by the specific adapter and should be referenced by the individual adapter documentation. The general format of the wrapper is explained in "Raw event normalization" on page 26.

Events are grouped by three identifying fields: baseID, sub1ID, and sub2ID. An event is only processed if an associated ECI is found to match these fields.

After an ECI matches a given raw event, a series of configuration tests are performed on the raw event to determine if the event should be processed further. Configuration test checks on a raw event include the following parameters:

Event Active	The event must be active in order to be processed.
Is Managed	Un-managed objects are not processed.
‹ *x* events in *y* minutes	Sliding window squelch.
ASL hook	Hard-coded test of other values.

"Setting up ECI configuration parameters" on page 32provides details on all the available ECI configuration values. Failure of any raw event test prevents the ECI process from determining what unpublished notification will be created or updated.

After basic event tests are complete, the event is formed into a notification. This phase of processing allows the instance, class, event name and all notification values to be changed based upon event data with variable substitution from raw event data. After this point, it is an unpublished notification that is no longer connected to the raw event, though the event data may be passed in the notification.

NOTICE

ECI configurations are saved as *.ncf files and remain unchanged through the stopping and starting of the Adapter Platform (OI) server.

There are several ways in which ECI objects can be loaded or added into a SAM/Adapter Platform server. ECI objects can be:

- Defined by *.ncf files provided from other sources (that is, pre-packaged configurations).
- Created directly from the EMC Smarts NOTIF Editor from the Edit menu.
- Merged into the EMC Smarts NOTIF Editor based on specifications in the trap_mgr.conf file.

Additionally, the EMC Smarts NOTIF Editor provides a means to copy an existing ECI object as the basis for another, and to save all defined configurations to an .import file.

After a raw event arrives and the best ECI object match is found, the behavior of the configuration object is performed relative to the event. If ECI objects are modified with the EMC Smarts NOTIF Editor, the effects of the modifications take place when the next instance of the relevant event arrives.

Raw event normalization

Raw events are passed to EMC Smarts NOTIF through adapters after being *normalized* into a common structure. This structure provides the required basic event information and additional data used for further processing. The format of the data stream is a pipe-separated ("|") string as shown below:

```
<eventTimeStamp>|<eventAgent>|<eventBaseID>|<eventSub1ID>|<eventSub2ID
   >|<Keyword1|Value1>|
   <Keyword2|Value2>.....
```

The pipe-separated data stream includes the following information:

- `eventTimeStamp` — Event occurrence time in UNIX time format; seconds since the Epoch
- `eventAgent` — Agent name or IP address
- `eventBaseID` — *dot*-separated hierarchical identifier string (abc.123.XYZ....) (for example, enterprise for SNMP)
- `eventSub1ID` — General identifier (generic trap number for SNMP)
- `eventSub2ID` — Specific identifier (specific trap number for SNMP)
- `Keyword1|Value1` — Keyword|value pair (OID1|value for SNMP)
- `Keyword2|Value2` — Keyword|value pair (OID2|value for SNMP)

The `eventBaseID`, `eventSub1ID`, and `eventSub2ID` parameters are used as the unique event identifiers to match with ECI objects. The number of `Keyword|Value` pairs is variable. Event IDs may use all three fields, the first two, or only the `eventBaseID`; it depends on the raw events being processed.

For specific adapters, the adapter documentation provides details to obtain information on the composition of the ID strings and the Keyword|Value pair mapping.

NOTICE

EMC Smarts NOTIF replaces pipe character ("|") in field.(varbind) by double dashes ("--") to process the raw event.

Pushing events into EMC Smarts NOTIF

EMC Smarts NOTIF uses a named queue to process incoming events. Events are normalized into a common structure and pushed into this queue through MODEL methods. The named queue for events is notifInterface::notifEventQueue. The method used to push an event into the queue is:

```
void pushEvent ( string encodedEvent, string sourceSystemId );
```

The `encodedEvent` is a pipe-separated ("|") string having the normalized format described in "Raw event normalization" on page 26. The `sourceSystemId` is a string identifying where this event came from.

The following example ASL file shows how a raw SNMP event is pushed into EMC Smarts NOTIF:

```
rawEvent.asl

START {
   .. eol
}
do {
   queue = object("notifInterface", "notifEventQueue");
   if(queue->isNull()) {
      print(queue . " was null!");
      return;
   }

   eventTimeStamp = time();
   eventAgent = "192.168.1.100";
   enterprise = "TestECI";
   generic = "";
   specific = "";
   oid1 = "testOID";
   oidValue1 = "test value";

   completeString = eventTimeStamp . "|" . eventAgent . "|" .
   enterprise . "|" . generic . "|" . specific . "|" . oid1 . "|" .
   oidValue1;
   queue->pushEvent(completeString, "rawEvent.asl");
}
```

ECI event matching

EventConfigInfo objects (ECIs) are the configuration objects that instruct EMC Smarts NOTIF how to handle incoming raw events. When a raw event arrives, the best matching ECI is identified and that is used to determine if a notification should be created, and if so, the instanceName and eventName to be associated with the notification.

ECI objects are named based upon a tree structure from the most generic to the most specific field, in the form of `eventBaseID_eventSubID1_eventSubID2`. An example would be:

Vendor.Product.Version.Equipment.Type_EquipmentSubType_SpecificType

SNMP defines ECI objects as Enterprise_Generic_Specific. Examples of how these names relate to events are given in the following section. Table 1 on page 28 describes the ECI naming rules.

Table 1 ECI naming convention rules

eventBaseID naming rules
• May be any number of *dot*-separated fields, the last of which may be *
eventSub1ID and eventSub2ID naming rules
• May be explicit, *, or null • If `eventSub2ID` is explicit or *, `eventSub1ID` must be explicit or * • `eventSub1ID` may exist with `eventSub2ID` null

Matching examples

When raw events are delivered to EMC Smarts NOTIF, EMC Smarts NOTIF finds the best match between a raw event and an ECI in its repository to handle that event. A *best match* is determined by first looking for an ECI that matches `eventBaseID_eventSubID1_eventSubID2` (in SNMP, these fields are referenced as enterprise, generic, and specific fields). If the original event is not matched by an exact name for these parameters, wildcard evaluations are performed in order from right to left.

Examples of how EMC Smarts NOTIF matches raw events to ECI objects are:

Defined ECIs

1. ECI-ABC.123.XYZ_TEST_EVENT
2. ECI-ABC.123.XYZ_TEST_*
3. ECI-ABC.123.*_TEST_*
4. ECI-ABC.*_*_*
5. ECI-ABC.*_*
6. ECI-ABC.*

Raw event (`eventBaseID_eventSub1ID_eventSub2ID`)	ECI processing
ABC.123.XYZ_TEST_EVENT	Uses ECI 1 — Exact match for all three `eventBaseID_eventSub1ID_eventSub2ID` fields
ABC.123.XYZ_TEST_FOX	Uses ECI 2
ABC.123.xyz_TEST_EVENT	Uses ECI 3
ABC.123.xyz_CAT_EVENT	Uses ECI 4
ABC.123.XYZ_TEST	Uses ECI 5 — Does not match ECI 4 since the `eventSub2ID` field is null
ABC.123.XYZ.wxy.zero	Uses ECI 6 — Does not match ECI 4 or ECI 5 since the `eventSub1ID` and `eventSub2ID` fields are null

Designing ECI naming conventions

Although the naming of ECI objects is determined by the designer of the adapter, it is worth noting here as the entire structure of the ECI tree is based on how this is accomplished. As an example, assume that events are built based upon the date. Some events may use the time format *mmm/dd/yyyy*, others may use *yyyy/mmm/dd*. If the former is used, (*mmm/dd/yyyy*), the month shows up as the primary ID (`eventBaseID`), and the EMC Smarts NOTIF Editor branch shows based on month:

Raw event wrapper of IDs

DATE.Apr_01_1996

DATE.Apr_01_1997

Specific defined ECIs

ECI-DATE.Apr_01_1996

ECI-DATE.Apr_01_1997

EMC Smarts NOTIF Editor tree view

Apr -

01 -

1996
1997
1998
02 -

...

A single ECI may be defined to process any month of any year on a specific day by creating `ECI-DATE.*_01_*`

However, a common ECI cannot be created to process *any* day of *any* year given this wrapper format. If the wrapper format was *yyyy/mmm/dd,* then the tree would appear as:

EMC Smarts NOTIF Editor tree view

1996 -

Apr -

01
02
...
1997 -

Apr -

01
02
...
1998 -

Apr -

...

In this case, an ECI could be `ECI-DATE.*_*_01` to process any event on the 1st:

`ECI-DATE.*_Apr_*` to handle any event in April.

`ECI-DATE.1996_*_*` to handle any event in 1996.

`ECI-DATE.*_*_*` to handle any event.

There are two issues from the previous example, first is the view from the EMC Smarts NOTIF Editor where it is undesirable to have hundreds of ECI objects showing under one branch. Second is the selection of what fields are used for the `eventBaseID`, `eventSubID1`, `eventSubID2` fields greatly impact how ECI objects are designed.

ECI notification naming

The ECI object is responsible for creating the unpublished notification that will be processed by the NCI thread for further processing. The primary keys for NCI matching are the instance ClassName and EventName, both of which are determined in the ECI as shown in Figure 8 on page 35.

Naming of the ECI is independent of the naming of the notification created by the ECI but consistency is important to avoid confusion. For example, `ECI-EventABC` could create a notification with an event named `EventXYZ`. Although this is possible, it is not desirable because the connection between an ECI and an NCI will not be apparent. Avoid inconsistent translations of event names to notification names.

A feature of the NCI name matching allows for the use of a wildcard at the end of the event name, for example: `Down-on-port-*`. It is important to construct the notification event name such that variables are placed at the end of the event name string. `Down-on-port-5` may use the above event name in the NCI for a match whereas `Port-5-Down` does not facilitate an NCI wild carding match.

Enumeration of keyword/value pairs

After EMC Smarts NOTIF has determined the appropriate ECI for a raw event, but before it starts processing, it will see if the values for the keyword/value pairs should be translated based on enumeration tables. For example, a keyword/value pair with `OID = .1.3.1.6.2.9.4432.12` may have a table that permits incoming values to be translated. The value 1 may mean *off* and the value 2 may mean *on*. In general, if an object ID (OID) has a translation table, EMC Smarts NOTIF converts the value supplied in the raw event to the translation value specified by the table.

In some situations, some devices may append additional nodes to the OID. For example, a device appends an IP address to the OID so that a given raw event might be .1.3.1.6.2.9.4432.12.192.168.1.234. EMC Smarts NOTIF allows you to flag an ECI as one which might have pairs that could use wildcard enumeration lookups. What this means is that EMC Smarts NOTIF will first see if there is an enumeration lookup table for .1.3.1.6.2.9.4432.12.192.168.1.234, and if none is found *and* wildcards are possible for the ECI, then EMC Smarts NOTIF would look for enumeration mapping tables for .1.3.1.6.2.9.4432.12.192.168.1, .1.3.1.6.2.9.4432.12.192.168, .1.3.1.6.2.9.4432.12.192, .1.3.1.6.2.9.4432.12, and so on till a lookup table is found or the name runs out of *dot*-separated values. This is a relatively computationally intense process, so EMC Smarts NOTIF also permits the administrator to set a maximum number of trailing nodes to check before giving up. In our example, where some devices are known to append IP addresses, the administrator might set the maximum number of trailing nodes to be four.

To summarize wildcard enumeration lookups:

- At the system level (in the configuration file) indicate the maximum number of nodes to drop in checking for enumeration tables.
- For specific ECIs that are used with raw events that might have OIDs that need wildcard lookups, set the ECI `Use Enum Wildcard` flag to `true`.

Note this example is specific to SNMP, but any dot-separated keyword string will work in the same manner. A keyword/value pair may appear as `Smarts.textSeverity = Critical` and be processed by an enumeration of:

Critical -> 1

Major -> 2

....

The display name of enumerated values may be changed to make the EMC Smarts NOTIF Editor tree view more readable. For example, an enumerated value of .1.3.6.1.4.1.9.9.37.2.13 may not have much meaning to the reader, but typing a display name of Trap.Cisco.FRU.Status makes it easy to identify the actual use of the enumerated value.

Value substitutions

The real-time substitution mechanism begins from the left of the string and finds the first substitution instructions. It evaluates that substitution and comes up with a new version of the string and starts again to evaluate from the left. Nesting of substitution strings may become complex and care should be taken to confirm desired results through testing. An example of using substitution values would be configuring the severity of an ECI to be `$V2$`, that is, the severity is set to the value of the second keyword/value pair. The severity of an ECI may also use something like `$extract(extract(V2,"[",3),"=",1)$`, but this requires a full understanding of the input string and substitution use. A description of the scripting system used within the substitution feature and a complete list of substitution values are listed in Appendix A, “Substitution Scripting and Values.”

Setting up ECI configuration parameters

After a matching ECI is identified for an incoming raw event, the configuration values for that ECI are used to process the event. You can set the configuration values for an ECI through the following tabs in the ECI Setup dialog box:

- General
- Event Setup
- Event Options
- User Defined
- Calc’d Values

This section provides screen shot examples of an ECI setup and descriptions of each field.

General values

The General tab of the ECI Setup dialog box shown in Figure 6 on page 32 defines the basic behavior of the raw event.

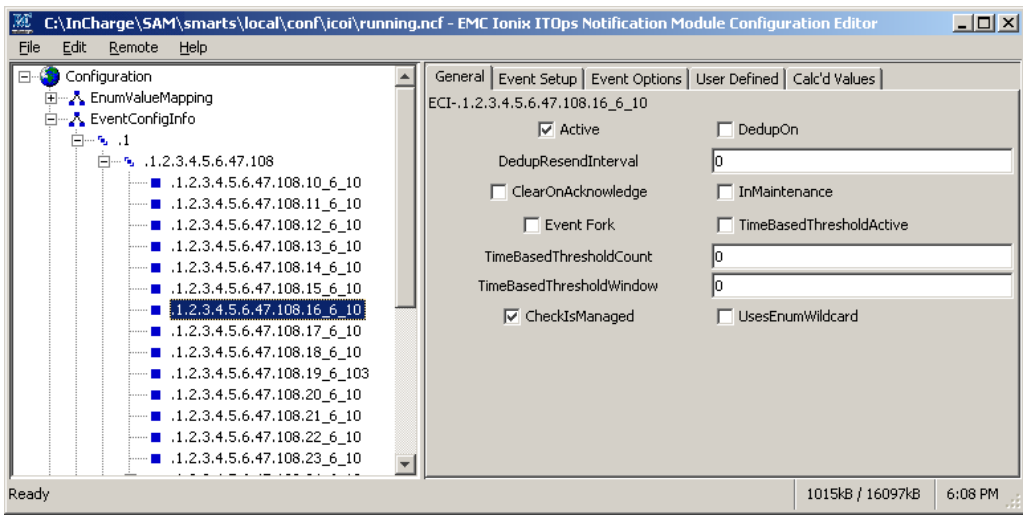


Figure 6 General tab of the ECI Setup dialog box

The fields in Figure 6 on page 32 are:

- **Active** — Selecting this checkbox means that the corresponding raw alarm is to be handled.
- **DedupOn** — Selecting this checkbox means that the corresponding event will always be filtered unless the notification has cleared. If the DedupOn checkbox is not selected, every occurrence of the event will be forwarded with no filtering for deduplication.
- **DedupResendInterval** — If this value is greater than zero and the DedupOn option is selected, then the event will be resent if the time since the event was last published is greater than this number of seconds. For example, if the DedupResendInterval is 60, even if there is already one of these events, if it is more than 60 seconds old, it will be resent.
- **ClearOnAcknowledge** — Selecting this checkbox causes the resulting notifications to be cleared when acknowledged.
- **InMaintenance** — Selecting this checkbox means the event occurred in maintenance.
- **Event Fork** — Selecting this checkbox means that the raw matching event will also look for ‹*ECI-name*›-2 to create a second thread for additional processing. If the ‹*ECI-name*›-2 has Event Fork selected, EMC Smarts NOTIF will then look for ‹*ECI-name*›-3, and so on.
- **TimeBasedThresholdActive** — This checkbox should be selected if the values for TimeBasedThresholdCount and TimeBasedThresholdWindow are to be used for x times in y seconds time-based event thresholding (where x is the TimeBasedThresholdCount value and y is the TimeBasedThresholdWindow value). For example, select the TimeBasedThresholdActive checkbox if you want a particular event sent if it occurs three times in a 15-second window.
- **TimeBasedThresholdCount** — The number of events that occurs in a given amount of time (that is, the TimeBasedThresholdWindow value) that triggers the forwarding of a particular event. For example, if the TimeBasedThresholdCount value is 3, and the TimeBasedThresholdWindow value is 15, then that means the event should be sent if there are 3 of these events that occur in 15 seconds. This enables you to filter out the odd once-in-a-while event.
- **TimeBasedThresholdWindow** — The number of seconds in which a particular number of events (that is, the TimeBasedThresholdCount value) occur that triggers the forwarding of a particular event. The TimeBasedThresholdWindow is a sliding window; for example, event 1 that occurs at 1 second, event 2 that occurs at 14 seconds and event 3 that occurs at 16 seconds will not generate a notification since all three of these events did not occur within a 15-second window, but event 4 that occurs at 18 seconds will create a notification because events 2, 3, and 4 occurred within a 15-second window.

  The DedupOn feature can be used with time-based event thresholding to prevent multiple notification updates. To explain how deduplication and time-based event thresholding work together, consider the example in which there are 3 events of the same type in 15 seconds, but the events are all deduplicated (that is the DedupOn checkbox is selected for this event type).

The underlying EMC Smarts NOTIF behavior would be:

1. The event is triggered the first time and ignored (since it has not yet hit the 3 events in 15 seconds threshold requirement).
2. The second and third events are deduplicated before they trigger the time-based thresholding rule.

NOTICE

To deduplicate this event and have it trigger the threshold would require the event to come in an onset, then clear, then onset, then clear, and finally one last onset and clear. The reason is that the deduplication will deduplicate any event coming in unless it is a clear.

Time-based thresholding generally has no clear event so the ClearOnAcknowledge feature described above and/or the Expiration feature described in the Event Setup tab should be used to clear the notification.

NOTICE

The ClearOnAcknowledge feature works at the SAM level, but does not by default clear the event if it came from the Adapter Platform. This can create confusion between notifications in the console between the Adapter Platform and SAM.

- **CheckIsManaged** — Selecting this checkbox means that the IsManaged property of the instance for this event must be true in order for this event to be processed. If the IsManaged property of the instance for this event is false, then the event will be logged and dropped.
- **UsesEnumWildcard** — Selecting this checkbox means that this ECI might have keyword/value pairs that may use wildcard enumeration lookups.

NOTICE

"Enumeration of keyword/value pairs" on page 31 provides more information on this feature.

A useful tip for navigating through the EMC Smarts NOTIF Editor is that the description of a specific configuration parameter will display when your mouse hovers over a checkbox or data entry field in the EMC Smarts NOTIF Editor. Figure 7 on page 35 shows an example of

when the mouse-hover tool is used for the Event Fork checkbox for a particular ECI object. You can also use the mouse-hover tool for NCI object EMC Smarts NOTIF Editor parameters.

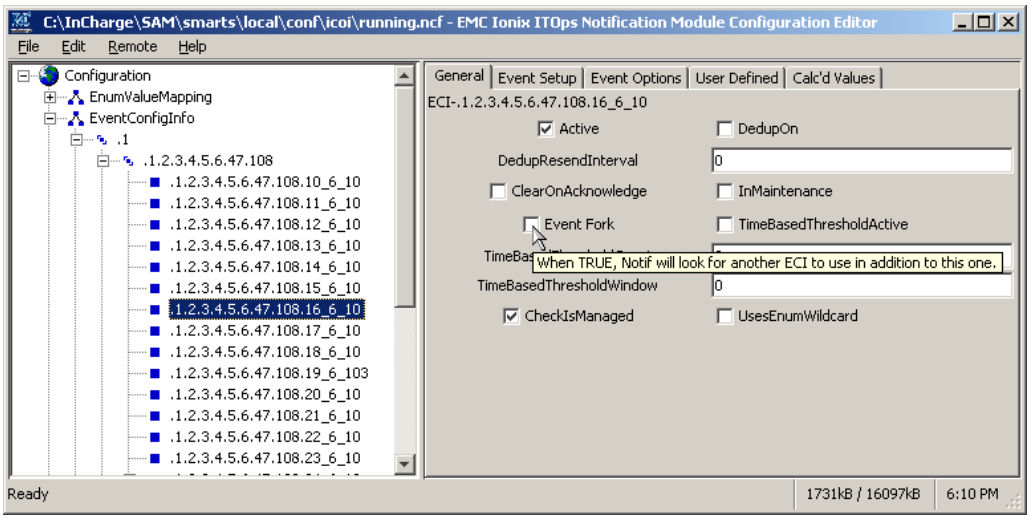


Figure 7 Example of mouse-hover tool in the EMC Smarts NOTIF Editor

Event Setup values

The Event Setup tab of the ECI Setup dialog box shown in Figure 8 on page 35 defines the properties of the notification created by the raw event.

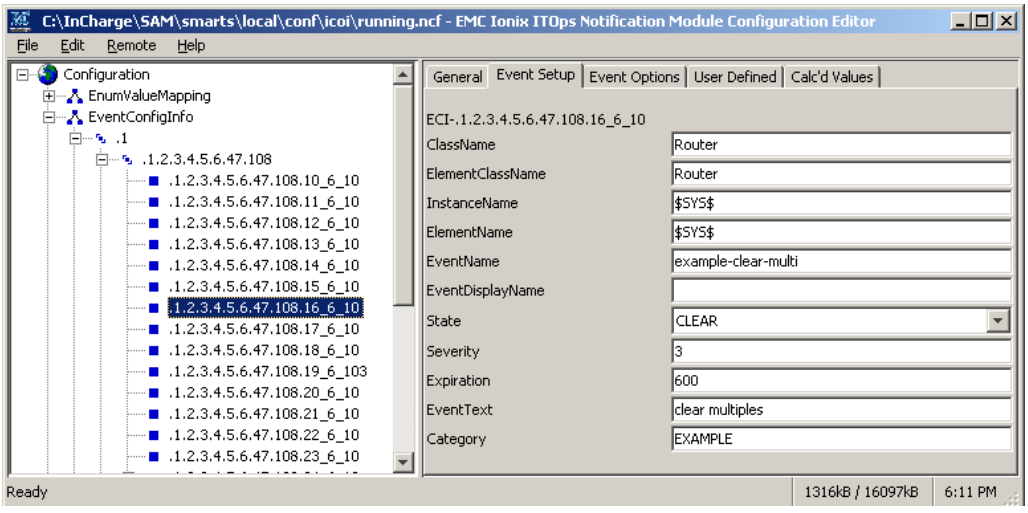


Figure 8 Event Setup tab of the ECI Setup dialog box

The fields in Figure 8 on page 35 are described below:

- **ClassName** — The class of the instance against which the notification is posted. This includes substitution values.
- **ElementClassName** — The class of the element. This includes substitution values.
- **InstanceName** — The name of the instance. This includes substitution values.
- **ElementName** — The name of the element. This includes substitution values.

- **EventName** — The name of the event used in the notification. This includes substitution values.
- **EventDisplayName** — The name of the event displayed in the EMC Smarts Notification Log Console.
- **State** — The state used for the notification, which is either `NOTIFY` or `CLEAR`.
- **Severity** — The numeric EMC Smarts value or substitution value.
- **Expiration** — The length of time in seconds the notification will remain active.
- **EventText** — The text of the notification, which includes substitution values.
- **Category** — The category of the event used in the notification.

Event Options values

The Event Options tab of the ECI Setup dialog box shown in Figure 9 on page 36 defines additional event options for the notification.

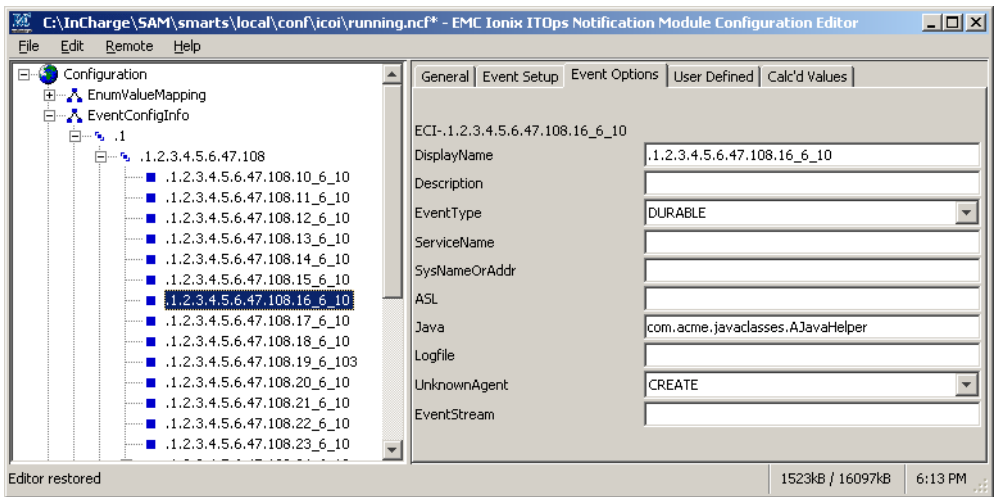


Figure 9 Event Options tab of the ECI Setup dialog box

The fields in Figure 9 on page 36 are described below:

- **DisplayName** — The display name of the event used in the notification.
- **Description** — The description of the event used in the notification.
- **EventType** — The type of event used in the notification, either `DURABLE` or `MOMENTARY`.
- **ServiceName** — The name of the service used in the notification.
- **SysNameOrAddr** — The system name or IP address associated with the event used in the notification.
- **ASL** — The ASL script called before the posting of this notification.

NOTICE

EMC Smarts NOTIF uses the value stored in ASL to determine which file to load. The named ASL script is invoked from the local/rules or /rules directories. For example, if you type icoi-trapd/sample_hook.asl in the ASL field in the EMC Smarts NOTIF Editor,

EMC Smarts NOTIF will look up local/rules/icoi-trapd/sample_hook.asl and rules/icoi-trapd/sample_hook.asl. Appendix B, "ASL and Java Interfacing with EMC Smarts NOTIF" provides more information on how to write ASL scripts for use with EMC Smarts NOTIF event processing.

- **Java** — The Java class called.

  NOTICE

  Appendix B, "ASL and Java Interfacing with EMC Smarts NOTIF" provides more information on how to write Java classes for use with EMC Smarts NOTIF event processing.

- **Logfile** — The name of the log file recording information about this event.
- **UnknownAgent** — The action taken if the agent is unknown, either `CREATE` which will create the object, or `IGNORE` which will discard this event.
- **Event Stream** — When set, the processor guarantees that all events with a given event stream name will be processed in the order in which they were received. If the Event Stream option is not set, there is no such guarantee. An example of a scenario in which setting the Event Stream for an ECI would be helpful is the following: a particular network component is *flapping* and is generating onsets and clears for a single alarm very quickly. In this scenario, EMC Smarts NOTIF could receive an onset and then a clear within a few milliseconds. When the ECI does not have an Event Stream set, EMC Smarts NOTIF could schedule the onset into one processing queue, and the clear into another. Because it is multithreaded, EMC Smarts NOTIF could then possibly process the clear first, which would be incorrect. By providing an Event Stream on the ECI, the user is mandating EMC Smarts NOTIF to process the onsets and clears in order, and the problem is solved. Many ECIs can have the same Event Stream, and that guarantees that all of the incoming events for any of those ECIs are processed in the order in which they were received.

User Defined values

The User Defined tab of the ECI Setup dialog box shown in Figure 10 on page 38 defines additional configurable fields that are passed to the notification. Substitution values may be used in these user-defined fields. Both ASL and Java hook scripts can be initiated from EMC Smarts NOTIF processing, which can overwrite these and other notification values.

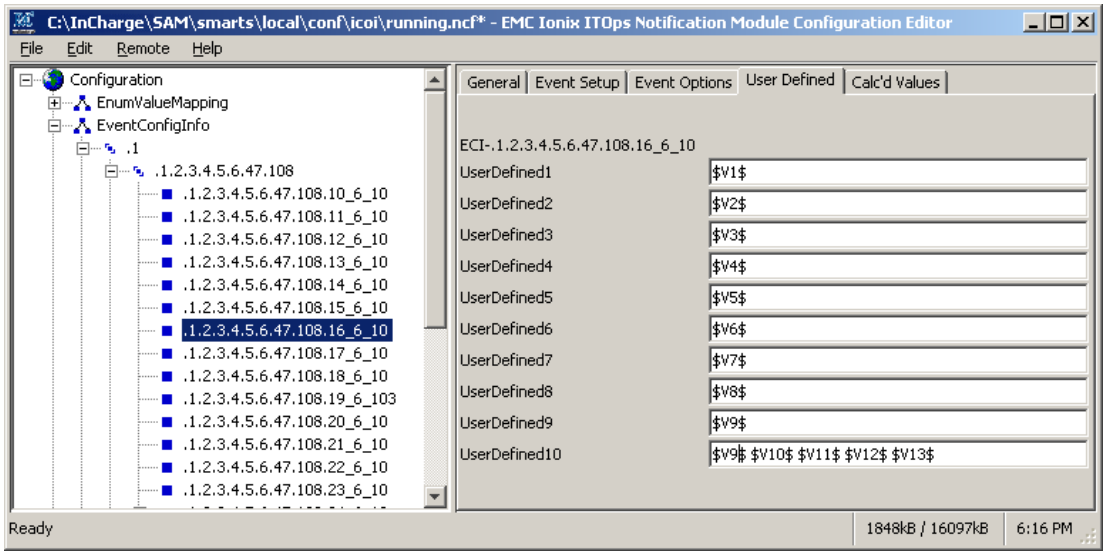


Figure 10 User Defined tab of the ECI Setup dialog box

Calculated values

The Calc'd Values tab of the ECI Setup dialog box shown in Figure 11 on page 38 defines calculated value instructions. The calculated values feature allows the user configuring EMC Smarts NOTIF to write simple scripts that use information from the event (or notification) being processed to calculate values for use in the EMC Smarts NOTIF processor. For example, a user could write a calculated value script that decides to discard an event when the value of SNMP trap varbind 3 (V3) is 0. Note that in the calculated values list in Figure 11 on page 38, literal values are enclosed with quotation marks (for example: "0") and variables are not (for example: V3).

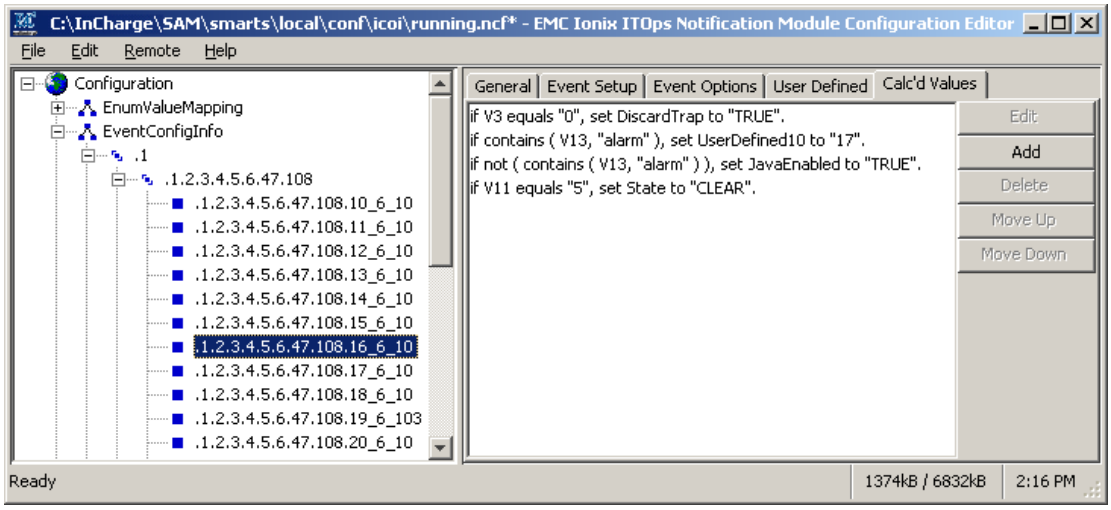


Figure 11 Calc'd Values tab of the ECI Setup dialog box

Calculated values comprises of a condition (value) (conditional) (value) and assignment (attribute) (value). Substitution values are allowed in any value field as shown in Table 3 on page 100 or select Help › Substitutions in the EMC Smarts NOTIF Editor.

Allowed conditionals include:

==, !=, ‹, ‹=, ›, ›=, contains, (left string contains right string), does not contain, (left string does not contain right string), starts with, ends with

NOTICE

== means equal, and != means not equal.

Legal assignment attributes include:

- ASLEnabled
- Category
- ClassName
- ClearOnAcknowledge
- DiscardTrap
- ElementClassName
- ElementName
- EventName
- EventText
- EventType
- Expiration
- InMaintenance
- InstanceName
- JavaEnabled
- Severity
- State
- UpdateUserDefinedFields
- UserDefined1 - 20

Examples of calculated values are the following:

- if V1 equals “1”, set Severity to “1”.
- if contains (SYS, “Demo”), set EventText to “Ignore events against”+SYS.

Adding calculated values

To add a calculated value to an ECI:

1. Select the ECI in the topology tree of the EMC Smarts NOTIF Editor.
2. Select the **Calc’d Value**s tab.

3. Click **Add**. The default **Edit Calculated Value** dialog box appears as shown in Figure 12 on page 40.

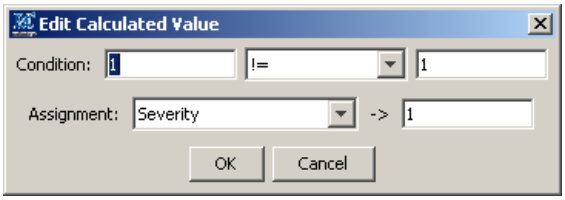
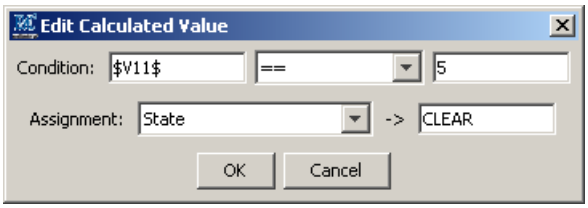


Figure 12 Edit Calculated Value dialog box

4. Type the appropriate values in this dialog box to create the calculated value for the ECI. For example:



NOTICE

In the user-editable fields in the Edit Calculated Value dialog box, you must include dollar signs for varbind variables; otherwise the variable will be read as literal text. For example, if V11 (instead of $V11$) was typed in the Condition: field in the above example, EMC Smarts NOTIF would read the V11 entry as a literal text string (not as the value of V11).
Also, the text typed for values in conditions and assignments should be in upper case. For example, State -> CLEAR, not State -> clear.

5. Click **OK**. The calculated value appears in the calculated value list as shown in Figure 13 on page 41.

NOTICE

In the calculated values list, literal values in the displayed calculated value are enclosed with quotation marks (for example: “5”) and variables are not (for example: V11).

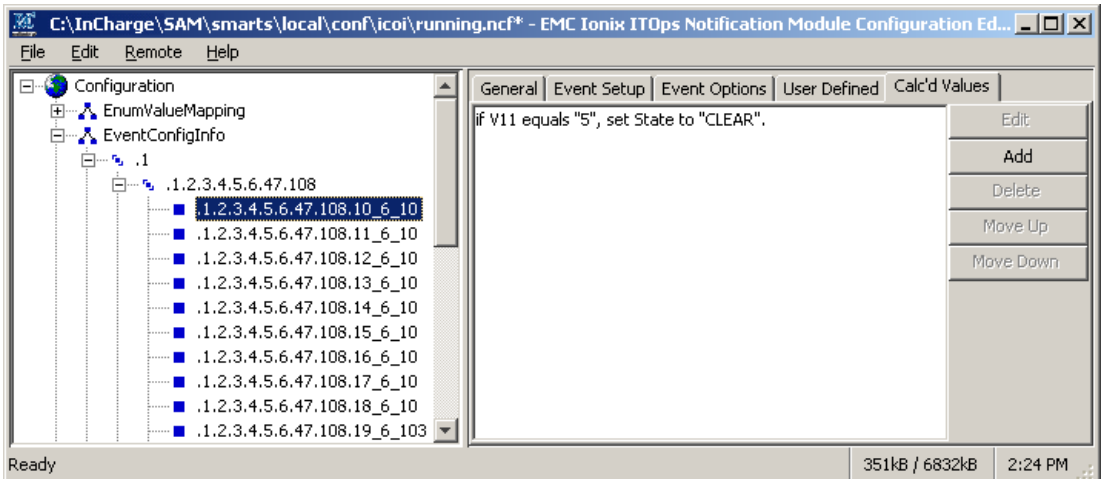


Figure 13 Example of adding a Calculated Value to an ECI

Editing calculated values

To edit a calculated value for an ECI:

1. Select the ECI in the topology tree of the EMC Smarts NOTIF Editor.
2. Select the **Calc'd Values** tab.
3. Select the calculated value you want to edit in the calculated value list.
4. Click **Edit**, or double-click the selected calculated value. The **Edit Calculated Value** dialog box appears as shown in Figure 14 on page 41.

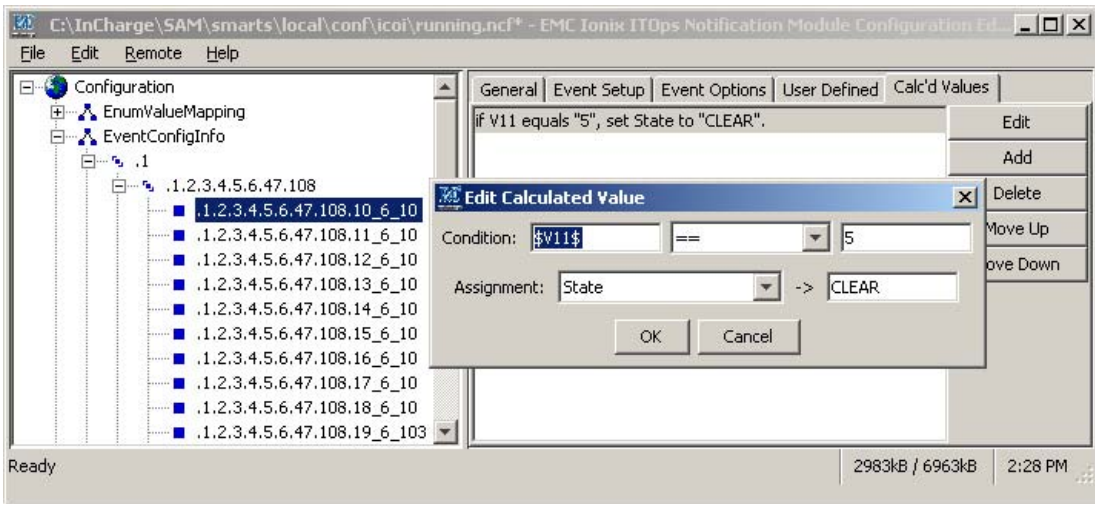


Figure 14 Example of editing a Calculated Value for an ECI

5. Type the appropriate values in the **Edit Calculated Value** dialog box to edit the calculated value for the ECI.
6. Click **OK**. The edited calculated value appears in the calculated value list.

When editing calculated values, consider the following:

- Calculated value instructions are evaluated from first to last in order. Later instructions can reset values set in prior instructions. Therefore, the order in which the calculated value instructions appear in this list is important.
- Some assignments affect processing. If `ASLEnabled` is set to `false`, then the ASL script will not run, even if it is present. Similarly, if `JavaEnabled` is set to `false`, then the Java script will not run even if present.
- In the user-editable fields in the Edit Calculated Value dialog box, you must include dollar signs for varbind variables. Otherwise, the variable will be read as literal text. For example, if V11 (instead of $V11$) was typed in the Condition field, EMC Smarts NOTIF would read the V11 as a literal text string (not as the value of V11).
- Text typed for values in conditions and assignments in the Edit Calculated Value dialog box should be in upper case. For example, State -> CLEAR is an acceptable entry in the Edit Calculated Value dialog box. If the user types State -> clear in the calculated value assignment, EMC Smarts NOTIF will log an error.
- In the calculated values list in the right pane of the EMC Smarts NOTIF Editor, literal text is enclosed with quotation marks, and variables are not. For example:

  if V3 equals “4”, set Severity to “2”
- When assigning to a boolean field, EMC Smarts NOTIF converts string values that are used for boolean expressions in the following manner:

  1. Converts the string to lower case.
  2. Removes any leading or trailing whitespace.
  3. Uses `true` if the string is one of: `true`, `yes` or `on`, or if the string evaluates to an integer that is not zero.

Copying ECI configuration parameters

You can copy ECI objects to reduce configuration time. To copy an ECI object, right-click the ECI object in the topology tree in the left pane and select Copy as shown in Figure 15 on page 43. The original ECI object that you copy is referred to as the *source*. The new ECI object is referred to as the *target*. The source object's properties are copied into the target object. Everything is copied except the `DisplayName` attribute.

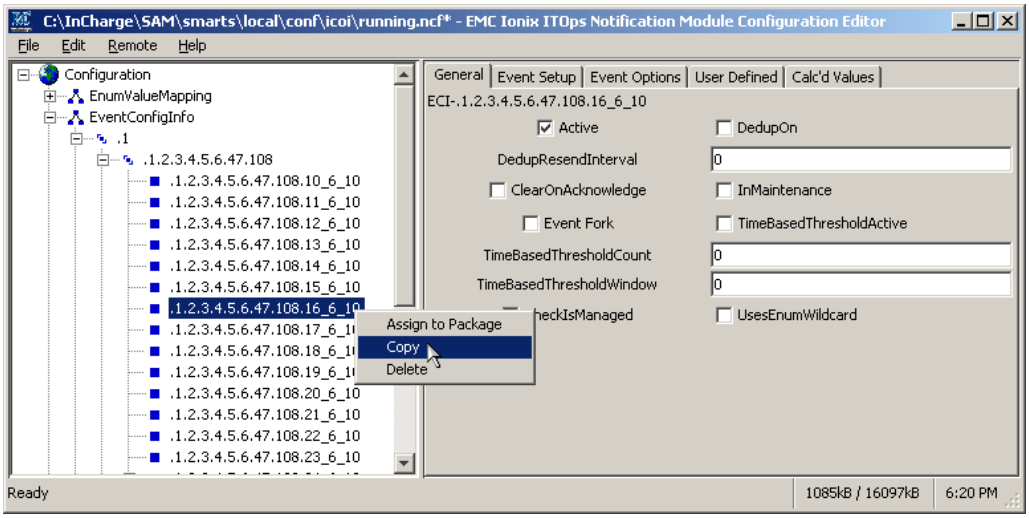


Figure 15 ECI copy feature

CHAPTER 3
Notification Configuration Information Processing

This chapter describes Notification Configuration Information (NCI) processing. This chapter consists of the following topics:

- NCI processing overview 46
- NCI notification matching 47
- Designing NCI naming conventions 50
- Setting up NCI configuration parameters 51
- Copying NCI configuration parameters 62

NCI processing overview

Notification Configuration Information objects (NCIs) are the configuration objects that instruct EMC Smarts NOTIF to handle incoming unpublished notifications. After EMC Smarts NOTIF completes the ECI processing phase, unpublished notifications are sent to the NCI portion of EMC Smarts NOTIF to process interactions between notifications. Figure 16 on page 46 shows the NCI processing steps.

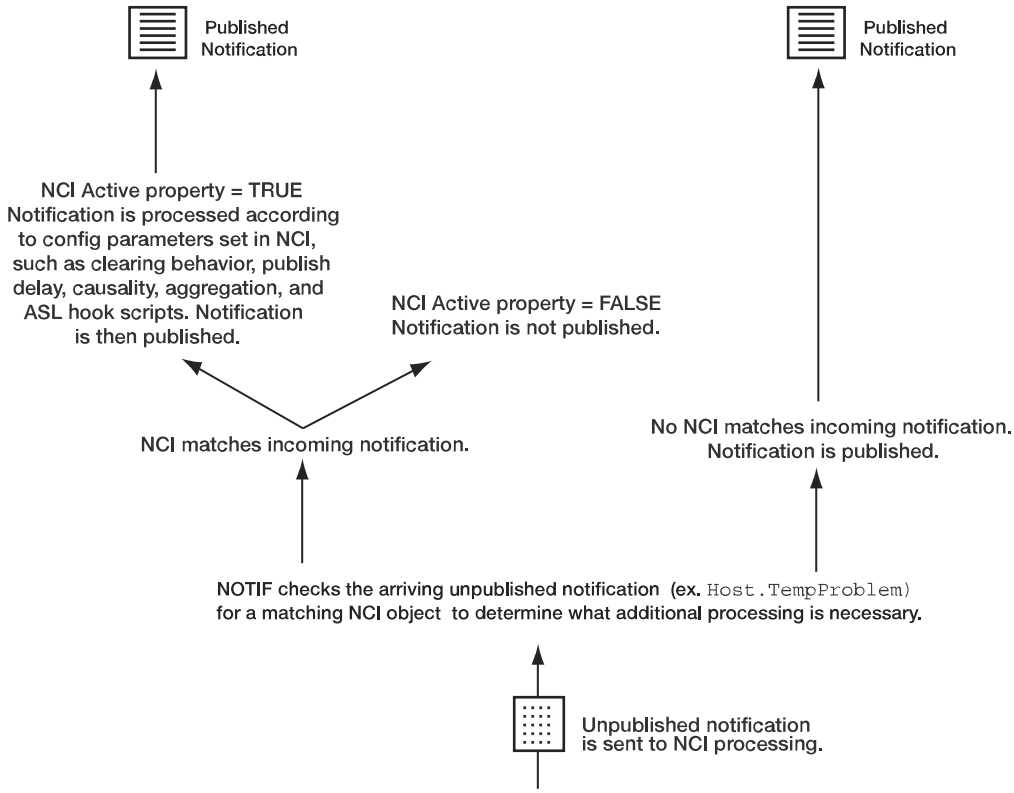


Figure 16 NCI processing steps

As shown in Figure 16 on page 46, when EMC Smarts NOTIF receives an incoming unpublished notification, it looks for a matching NCI object. If a matching NCI object is found for the incoming notification, EMC Smarts NOTIF checks several configuration parameters that are set in the NCI. If no matching NCI object is found for the incoming unpublished notification, the notification is published without further consideration.

When a matching NCI object is found for an incoming unpublished notification, the NCI processing portion of EMC Smarts NOTIF checks for the following configuration parameters:

1. Active property of the NCI object (that is, whether the Active checkbox is selected in the NCI setup EMC Smarts NOTIF Editor as shown in Figure 18 on page 52. If the NCI object is Active, EMC Smarts NOTIF continues to process that notification, checking the NCI configuration parameters listed below. If the NCI object is not Active, EMC Smarts NOTIF does not publish the notification.
2. "Clearing behavior" on page 51 — EMC Smarts NOTIF checks to see if the incoming notification clears other notifications, or inversely, whether this notification is cleared by other notifications.

3. "Publish delay" on page 54 — EMC Smarts NOTIF checks to see if there is a specified delay associated with the notification before further NCI processing continues.
4. "Causality" on page 54 — EMC Smarts NOTIF checks to see if this notification causes another notification, or inversely, if this notification is caused by another notification.
5. "Aggregation" on page 60 — EMC Smarts NOTIF checks to see if this notification aggregates from other notifications, or inversely, if this notification aggregates to another notification.
6. "NCI hook script" on page 61 — EMC Smarts NOTIF checks to see whether there is an ASL hook script that the NCI should call just before the notification is published (after all other NCI processing has been completed).
7. "Calculated Values" on page 62 — EMC Smarts NOTIF checks to see if there are any calculated values associated with the notification.

These NCI configuration values are used to process the notification. The NCI process of EMC Smarts NOTIF is an independent thread that receives information from other servers for the creation, deletion, and change of notifications through the ICS_NOTIFICATION hook script. It is not directly associated with ECI processing as shown in Figure 3 on page 18.

There are two reasons NCI objects exist within EMC Smarts NOTIF. They are either used to match incoming notifications, or they are used to create new notifications. In general, new notifications are those created because of aggregation.

NCI notification matching

When unpublished notifications are delivered to EMC Smarts NOTIF, it finds the best match between a notification and an NCI object in its repository to handle that notification. The notification must be active in order for it to be processed, that is, the Active checkbox in the NCI setup EMC Smarts NOTIF Editor must be selected as shown in Figure 18 on page 52.

Processing of the notification begins by looking for an exact match of class and event. A best match is determined by first looking for an NCI object that matches the `ClassName.EventName` of the incoming notification. If that does not match an NCI, EMC Smarts NOTIF will do the following:

1. EMC Smarts NOTIF will first look for an NCI that matches `ClassName.EventName*` (any ending). EMC Smarts NOTIF looks for an NCI that contains an asterisk (*) at the end to see if the EventName in the notification can match to the beginning of the EventName in the NCI (for example, the NCI named `Router.Down-*` would match with the notification named `Router.Down-2`).
2. If there is no NCI EventName with a wildcard to match the EventName in the incoming notification, EMC Smarts NOTIF then begins to climb the class hierarchy to determine if a superior class will match. At each step up the hierarchy, EMC Smarts NOTIF first checks for an exact event name match then a wildcard match. In this manner, a single NCI object can handle sub-classed objects as well as event names with different suffixes.

NOTICE

Note that EMC Smarts NOTIF expects either an exact event name match or a single wildcard match; ambiguities in wildcard matches will have indeterminate results as to which NCI configuration will be used for processing. A series of NCIs such as `Router.Test-1-1-*`, `Router.Test-1-*`, and `Router.Test-*` will have no determinate order; the first NCI found will be used with no further processing. Supporting the best match at the NCI EventName level has an exponential impact on processing speed and is not practical.

For the same reason, regex expressions are not supported within the NCI EventName string, only a trailing asterisk (*) is supported.

Figure 17 on page 49 illustrates the NCI notification matching process.

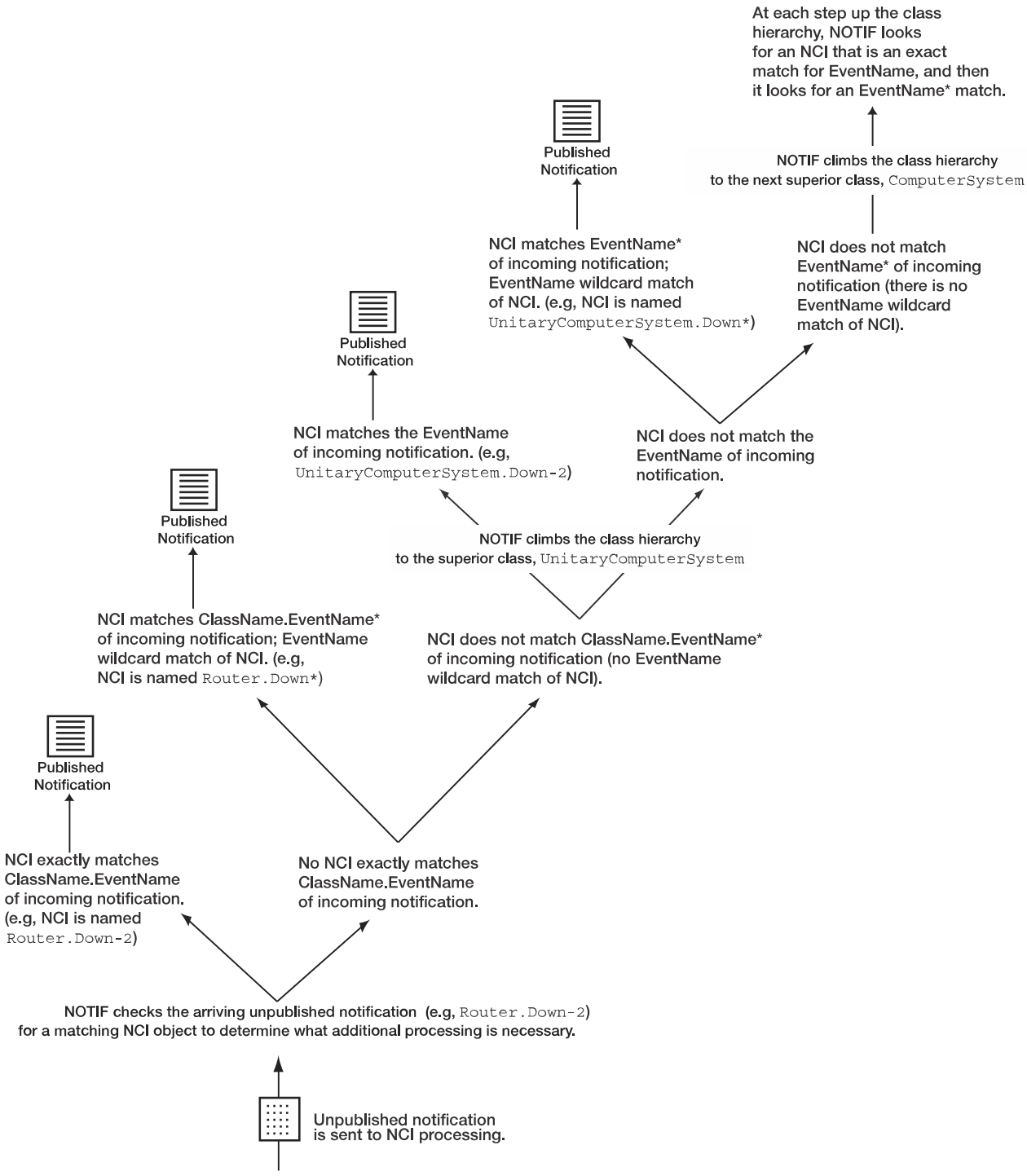


Figure 17 NCI notification matching process

Matching examples

NCI objects can be created for specific class instances or subclasses based on the overall design and wildcarded EventNames. Examples of how EMC Smarts NOTIF matches notifications to NCI objects are provided below:

Defined NCIs

1. XYZ-Router.Event-1
2. Router.Event-1
3. Router.Event-*
4. `UnitaryComputerSystem.Event-1`

Incoming notification (ClassName.EventName)	NCI processing
XYZ-Router and Event-1	uses NCI 1 — Exact match for `ClassName` and `EventName`
`ABC-Router` and `Event-1`	uses NCI 2 — `ABC-Router` is a router
`ABC-Router` and `Event-2`	uses NCI 3 — `ABC-Router` and `Event-2` match the wildcard EventName
`Host` and `Event-1`	uses NCI 4f

In the examples above, any subclass of router with an event name of `Event-1` will be processed by the NCI named `Router.Event-1` (that is, NCI 2), except for XYZ-Router which is a more specific router class definition. As a theoretical example, the NCI named `XYZ-Router.Event-1*` could represent the handling of any EventName other than that specified by `Router.Event-1`.

Designing NCI naming conventions

NCI objects are matched to incoming notifications based on the class and event name in the notification. The information in the notification is dependent on the underlying ECI process that created the NCI requirements. Therefore, the NCI requirements are determined in the configuration of the ECI object.

NOTICE

"ECI notification naming" on page 30 provides more information.

Setting up NCI configuration parameters

When setting up the configuration parameters for an NCI, there are six configuration parameters to consider:

- "Clearing behavior" on page 51
- "Publish delay" on page 54
- "Causality" on page 54
- "Aggregation" on page 60
- "NCI hook script" on page 61
- "Calculated Values" on page 62

Clearing behavior

ClearsForAllNotifications

An NCI can specify that it ClearsForAllNotifications as shown in Figure 18 on page 52. Selecting the ClearsForAllNotifications checkbox means that the given notification created for an event will clear all active notifications involving the same instance and element as the given event. When such a notification is created through an NCI, EMC Smarts NOTIF considers all the notifications related to the same instance (except the notification triggering this behavior), and clears them. The original specification in the Event Setup tab in the ECI determines if the triggering notification will be a `NOTIFY` or a `CLEAR` as shown in Figure 8 on page 35.

Consider the following example of the ClearsForAllNotifications functionality. There are three notifications (the Down, Faulty, and ActingUp notifications) for the Element::Instance named Interface::FaultyInterface. There is something wrong with FaultyInterface and the Interface is generating several different error alarms.

Assume that you have an NCI called `Interface.InterfaceIsGoodAgain` with ClearsForAllNotifications = TRUE. FaultyInterface may send a trap that results in a notification like `Interface_FaultyInterface_InterfaceisGoodAgain` being created when the interface finally fixes itself. This notification will trigger the NCI named `Interface.InterfaceIsGoodAgain`. The NCI ClearsForAllNotifications property will cause EMC Smarts NOTIF to clear all of the notifications against FaultyInterface. The `InterfaceIsGoodAgain` notification may or may not clear depending on its State (`NOTIFY` or `CLEAR`). Most users would have an ECI trigger `InterfaceIsGoodAgain` with a `CLEAR` state. The end result is that the device no longer shows active notifications on the console once the *all clear* trap is seen by EMC Smarts NOTIF.

Notifications on Interfaces other than FaultyInterface (since this is the Instance the `InterfaceIsGoodAgain` notification points to) would still remain active.

NOTICE

If EMC Smarts NOTIF is running on a SAM server, you must use the ClearsForAllNotifications option with caution. If this option is enabled, then all notifications against that particular instance (that is, router) are cleared and this may override the Caused/CausedBy relationship in the SAM codebook correlation.

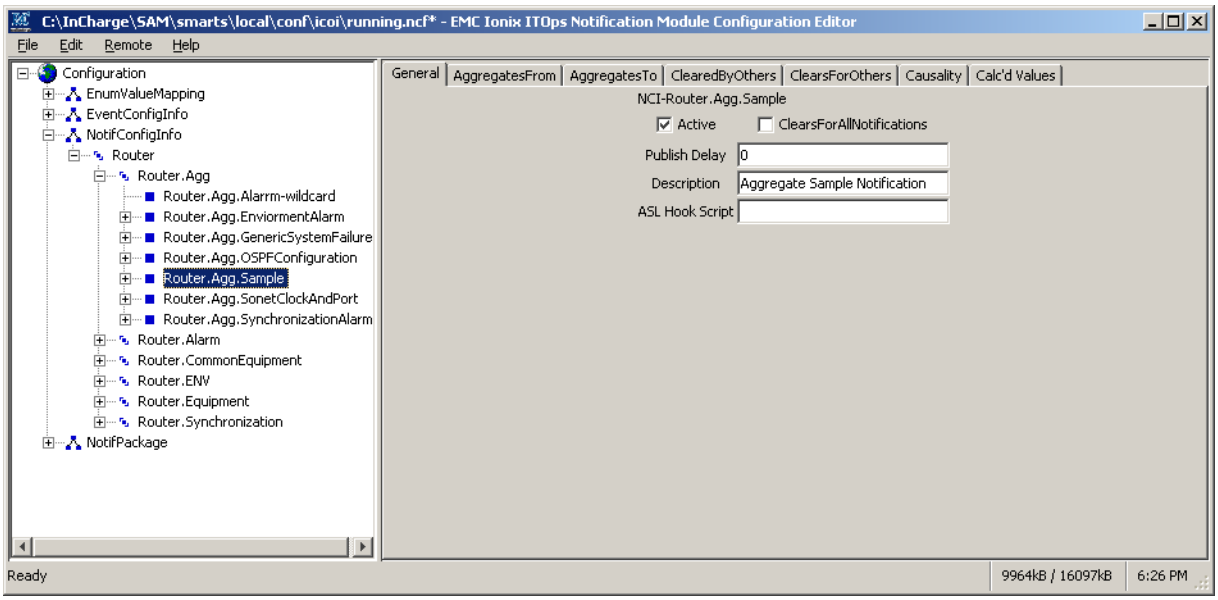


Figure 18 General tab of the NCI Setup dialog box

ClearsForOthers

You can specify more finely-tuned clearing of notifications in an NCI with the ClearsForOthers tab in the NCI Setup dialog box (see Figure 19 on page 53). The ClearsForOtherNotifications field in this tab allows you to specify one or more other NCIs that the given NCI clears for. After the specified notification is created, EMC Smarts NOTIF finds the existing notifications related to the same instance and with eventNames as specified in the ClearsForOtherNotifications setting. When the instance and event names

both match, the specified companion NCI(s) are cleared. For example, a `.Disk.Purged` notification may be configured to ClearsFor the `.Disk.UsageExceeded` and `.Disk.UsageWarning` notifications.

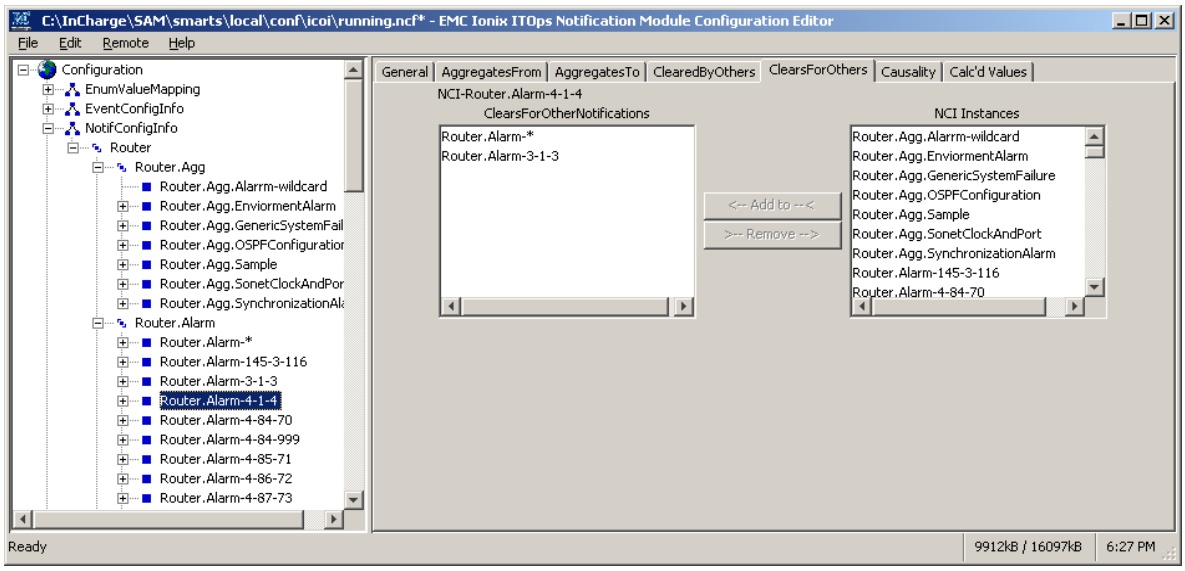


Figure 19 ClearsForOthers tab in the NCI Setup dialog box

ClearedByOthers

Inversely, a given NCI may be ClearedBy other notifications. In the ClearedByOthers tab, in the ClearedByOtherNotifications field, you can add all the notifications that clear this NCI. For example, as shown in Figure 20 on page 53, the NCI named `Router.Alarm-3-1-3` is cleared by the `Router.Alarm-4-1-4` notification.

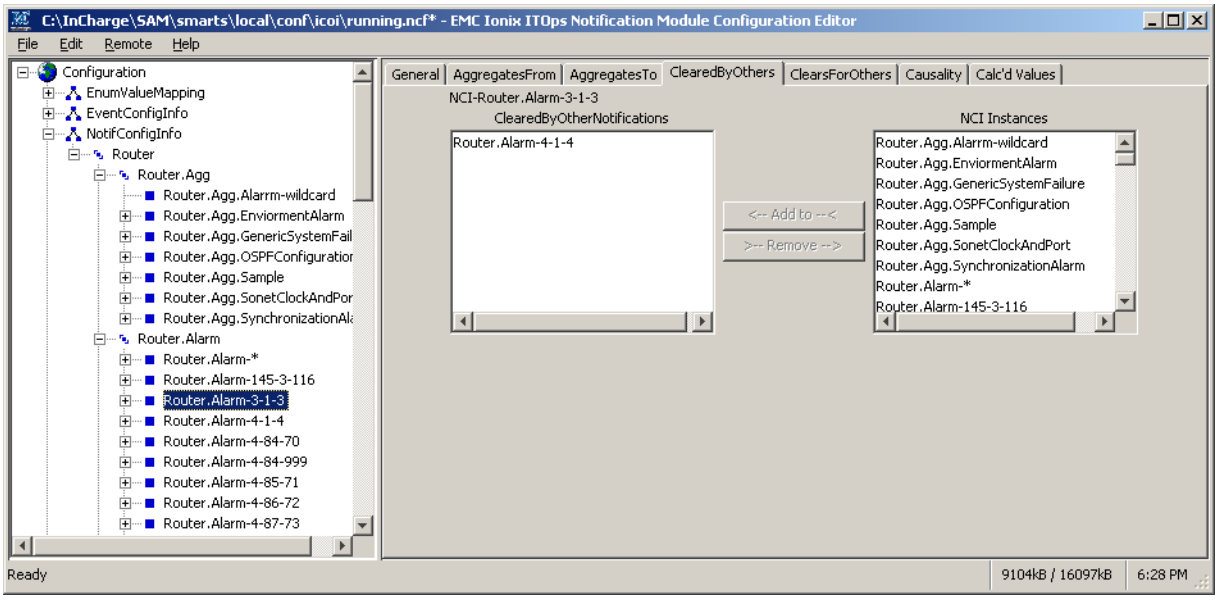


Figure 20 ClearedByOthers tab in the NCI Setup dialog box

Publish delay

After an NCI is found for an incoming notification, clearing behaviors are performed as discussed in the previous section. After any clearing is done, the notification may have a delay prior to checking Causes, Aggregates, and getting published. This delay is set in the NCI (in the Publish Delay field in Figure 18 on page 52) and is specified in seconds. For example, if you set a publish delay for 60 seconds, that means that the notification will be held for a minute before any further NCI processing is performed. If a clearing comes in during that minute, then the notification will be discarded.

Causality

You can configure an NCI to indicate that one notification causes another notification, or inversely, that one notification is caused by another notification. After EMC Smarts NOTIF checks an incoming notification for clearing behavior, and any publishing delay has expired, EMC Smarts NOTIF checks to see if the new incoming notification causes any other notifications or is caused by any existing notifications. If EMC Smarts NOTIF finds that the given notification causes or is caused by other notifications, EMC Smarts NOTIF relates the appropriate notifications.

You set up causes/causedby relationships in the Causality tab in the NCI Setup dialog box. For example, to set up a relationship where `NCI-Router.Alarm-145-3-116` causes `NCI-Router.Alarm-7-2-5`, you would do the following:

1. Click the `Router.Alarm-145-3-116` NCI object in the topology tree of the EMC Smarts NOTIF Editor.
2. Select the **Causality** tab.
3. Click **Add**.
4. The **Add Causes Relationship** dialog box appears, as shown in Figure 21 on page 54. Select `NCI-Router.Alarm-7-2-5` in the **Caused NCI:** drop-down list and click **Add**.

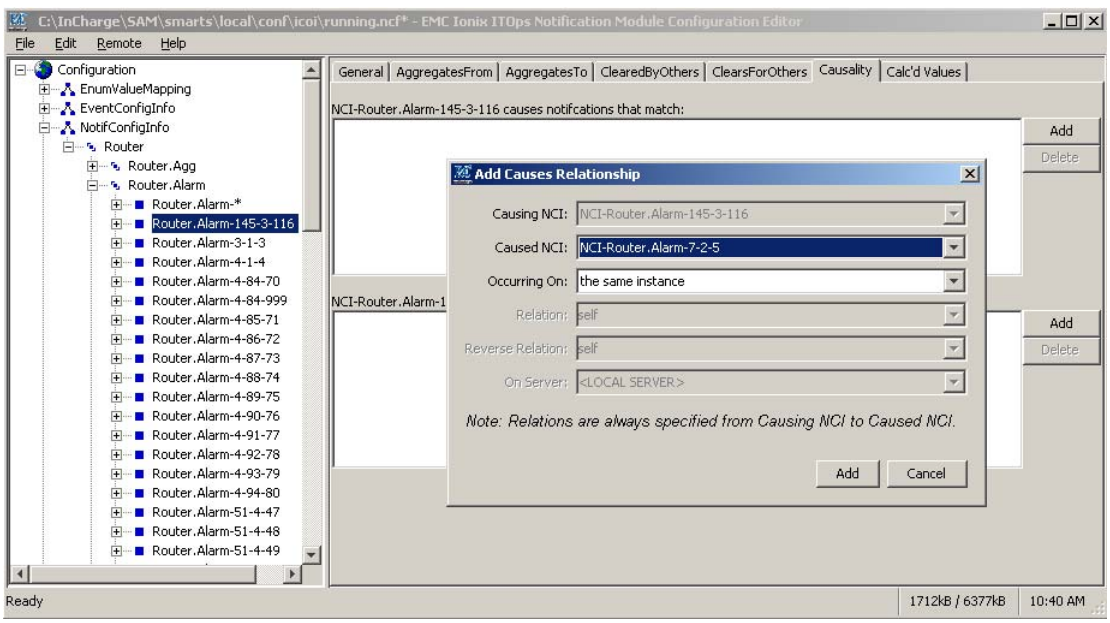


Figure 21 Add Causes Relationship dialog box

5. The **Causality** tab for `NCI-Router.Alarm-145-3-116` now shows that it causes `NCI-Router.Alarm-7-2-5`, as shown in Figure 22 on page 55.

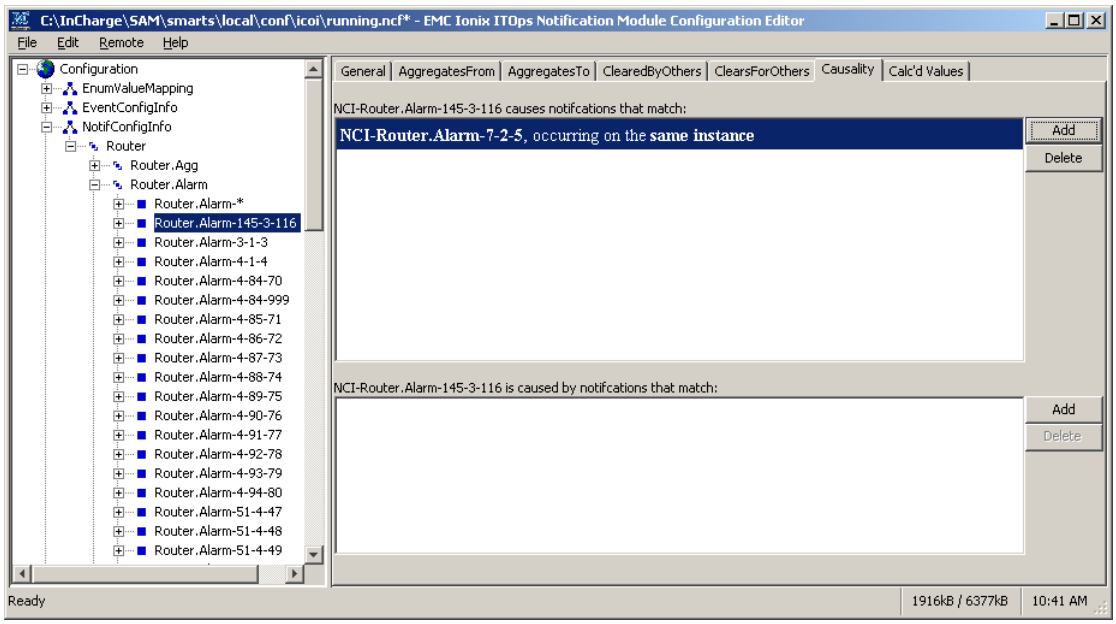


Figure 22 Causality tab shows NCI-Router.Alarm-145-3-116 cause NCI-Router.Alarm-7-2-5

When you click the `Router.Alarm-7-2-5` NCI object in the topology tree of the EMC Smarts NOTIF Editor, the **Causality** tab shows that this NCI is caused by `NCI-Router.Alarm-145-3-116`, as shown in Figure 23 on page 55.

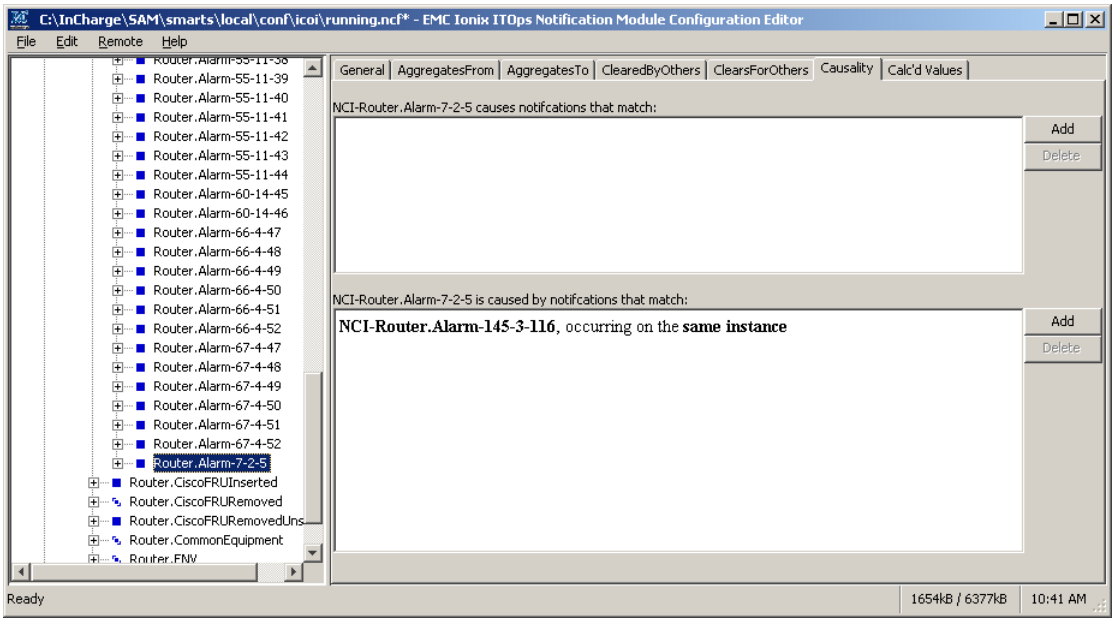


Figure 23 Causality tab shows NCI-Router.Alarm-7-2-5 cause NCI-Router.Alarm-145-3-116

In EMC Smarts systems, notifications usually occur on a particular instance within an element. For example, a Loss of Signal notification might occur on a Port instance within a Router element. EMC Smarts NOTIF can use a notification's instance and element to find other instances and elements, and then create causal relationships between the notifications on each of them.

EMC Smarts NOTIF traces causality through the EMC Smarts objects that correspond to the notifications. EMC Smarts NOTIF determines causality by tracing the topology objects in five ways:

- “Occurring on the same instance” on page 56
- “Occurring on an instance related to the notified instance” on page 56
- “Occurring on the same element” on page 57
- “Occurring on an instance related to the notified element” on page 58
- “Occurring on an element related to the notified element” on page 58

For example, in Figure 23 on page 55, the causality specification shows that the NCI named `Router.Alarm-7-2-5` is caused by the notification `Router.Alarm-145-3-116`. The relationship between these two notifications is “Occurring on the same instance.”

The causality specification shows all the other notifications caused by a given configuration, and the relationship between the given notification’s instance or element to other instances or elements. The relationship tracing allows notifications of a different class and instance to be related, that is, an interface down may be caused by a router down.

Occurring on the same instance

The “Occurring on the same instance” causality relationship in the EMC Smarts NOTIF Editor allows a notification on an instance to be set to cause another notification on the same instance. For example, consider the creation of a notification `PowerSupply::network:86.2.2.2:PowerSupply-1::Down` (note this has class, instance, and eventName elements). Assume the corresponding NCI indicates this notification causes `PowerSupply-1.NotRespondingToAgent` notifications. The `PowerSupply-1.Down` notification is set to cause the `PowerSupply-1.NotRespondingToAgent` notification on the same instance, `PowerSupply-1`.

Occurring on an instance related to the notified instance

This causality relationship means that using a given notification's instance and the designated relationship you can find all other candidate instances. If there are active notifications for any of these instances, those notifications will be related to the first notification.

For example, assume the `PowerSupply::network:86.2.2.2:PowerSupply-1::Down` notification has a corresponding NCI that indicates this notification causes `.Card.Down` notifications for any Cards this PowerSupply is Underlying. The creation of the notification and the NCI would have EMC Smarts NOTIF look for notifications whose instances are Cards and eventName is Down, and which the PowerSupply is Underlying the Card. If any are found, they are related—the `PowerSupply.Down` notification is set to cause the other notifications.

The above scenario describes one notification causing another. The inverse scenario is also possible where for example, a `.Card.Down` notification is caused by `PowerSupply.Down` and the causality is the same instance through the LayeredOver relationship (that is, the Card is LayeredOver the PowerSupply).

Another same instance example is shown in Figure 24 on page 57 where an `Interface.Down` notification on a particular interface causes an `Interface.Unresponsive` notification for another interface, and the relationship is ConnectedTo. In this example, the `Interface.`Down notification for the Router-A-Port-1 interface instance causes the `Interface.Unresponsive` notification for the Router-B-Port-5 interface instance.

When the `Interface::Router-A-Port-1::Down` notification and the corresponding NCI are created, EMC Smarts NOTIF finds the instance (`Interface::Router-A-Port-1`) and follows the ConnectedTo relationship. This finds the `Interface::Router-B-Port-5` instance with the `Interface.Unresponsive` notification, and the two notifications are related. The `Interface::Router-A-Port-1::Down` notification causes the `Interface::Router-B-Port-5::Unresponsive` notification.

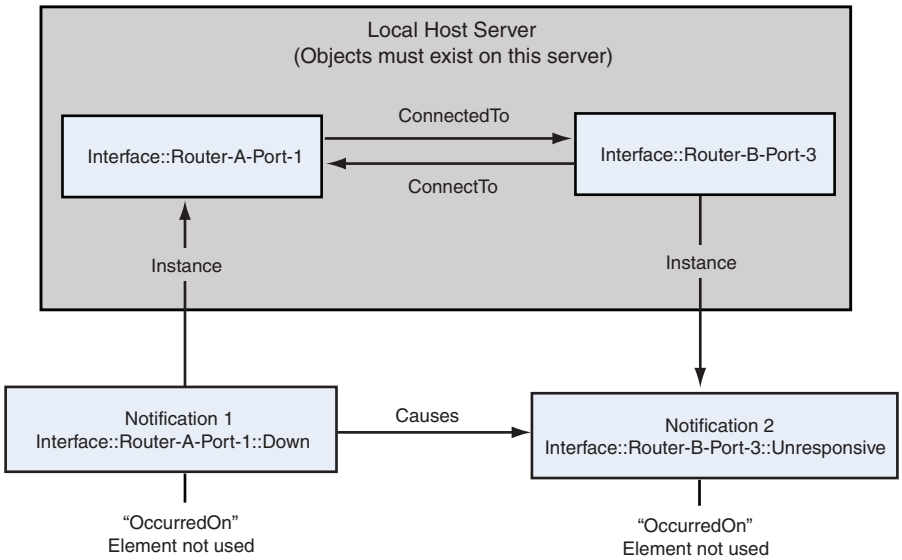


Figure 24 NCI causality relationship occurs on an instance related to a notified instance

Occurring on the same element

The "Occurring on the same element" causality relationship in the EMC Smarts NOTIF Editor allows a notification on an element to be set to cause another notification on the same element. For example, `PowerSupply.Down` may cause `Router.Down`.

Let us slightly change the preceding example where the NCI is for `PowerSupply::network:86.2.2.2:PowerSupply-1::Down`, but it is specified as "occurring on the same element" in the EMC Smarts NOTIF Editor. Assume the `.PowerSupply.Down` causes `.Router.Down`. If a `.PowerSupply.Down` notification is created, EMC Smarts NOTIF will find its elements (probably Routers) and if any of those routers had `.Router.Down` notifications, then those notifications would be related to the `.PowerSupply.Down` notification.

Occurring on an instance related to the notified element

The causes that flow from an element to an instance through a relationship are determined as follows:

1. Find the element for a given notification
2. Use the specified relationship to find related instances
3. See if there are corresponding notifications

As an example, the `PowerSupply::network:86.2.2.2:PowerSupply-1::Down` NCI is specified as "occurring on an instance related to the notified element" in the EMC Smarts NOTIF Editor and the relationship is ComposedOf. Assume the `.PowerSupply.Down` Causes `.Card.Down`. If a `.PowerSupply.Down` notification is created, EMC Smarts NOTIF will find its element (probably a Router) then follow the ComposedOf relationship. This would find all the Router's Cards and if any of those cards had `Card.Down` notifications, then those notifications would be related to the `.PowerSupply.Down` notification.

The advantage of this specification is that several objects within the element may be able to transfer causality without being directly related.

Looking at this causality in the inverse direction, the `Card.Down` NCI would show that relationship is followed in the instance to element direction and the relationship would be PartOf. When a `Card.Down` notification is created, it would follow its PartOf relationship to find an element, then look for all the notifications associated with that element, looking for a `.PowerSupply.Down`.

NOTICE

The "occurring on an instance related to the notified element" specification can be very processor-intensive for environments with large numbers of active notifications. Therefore, it is recommended that use of this specification be kept to a minimum.

Occurring on an element related to the notified element

Causes that flow from one element to a related element through a relationship are determined as follows: for a given notification, find its element. Then use the relationship to find another element. The notifications associated with the second element are examined for matches to the specified target notifications.

An example of an "occurring on an element related to the notified element" causality relationship is shown in Figure 25 on page 59 where an `SNMPTrap.Down` notification on a particular host causes an `SNMPTrap.Unresponsive` notification on a particular software service, and the relationship is HostsServices.

In this example, the `SNMPTrap::DummyHost::Down` notification causes the `SNMPTrap::DummySoftware::Unresponsive` notification. When the `SNMPTrap.Down` notification is created, EMC Smarts NOTIF finds its element (the Host named `test-host-1`) and follows the HostsServices relationship. This finds the SoftwareService element named `test-software-1`, and any SNMP traps associated with this

SoftwareService element that have `SNMPTrap.Unresponsive` notifications. In this way, the `SNMPTrap::DummySoftware::Unresponsive` notification is related to (caused by) the `SNMPTrap::DummyHost::Down` notification.

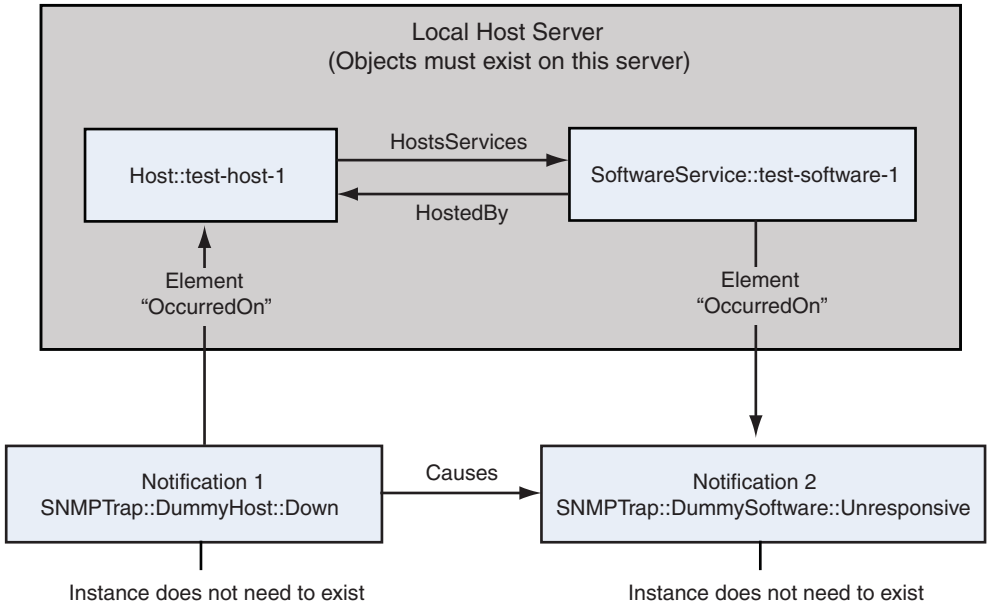


Figure 25 NCI causality relationship occurs on an element related to a notified element

Aggregation

Notification aggregation occurs when one or more notifications on the same instance trigger the creation of an Aggregate notification, which represents the summary of a complex problem. An Aggregate notification sets a notification based on other events being active.

For example, assume an NCI is created where `.NetworkConnection.Erratic` is specified and it AggregatesFrom:

.NetworkConnection.HighBitErrorRate,

.NetworkConnection.LostPacketsExceeded

.NetworkConnection.ExcessiveFramingErrors

Aggregation is useful when any number of notifications indicate a common problem. The severity of the aggregate notification is that of the highest severity of the underlying notifications. When all underlying notifications have cleared or expired, the aggregate itself will clear. This *roll-up* effect is applicable only from a EMC Smarts NOTIF Editor perspective, the individual notifications will still have the `isRoot` attribute set to `true`. Aggregation is limited to notifications against the same class and element.

Notifications may be accessed either through the AggregatesFrom tab or the AggregatesTo tab as shown in Figure 26 on page 60 and Figure 27 on page 61. However, the Event Text field is only available in the AggregatesFrom tab. In this field you can type the text of the notification and you can provide substitution values for user-defined values.

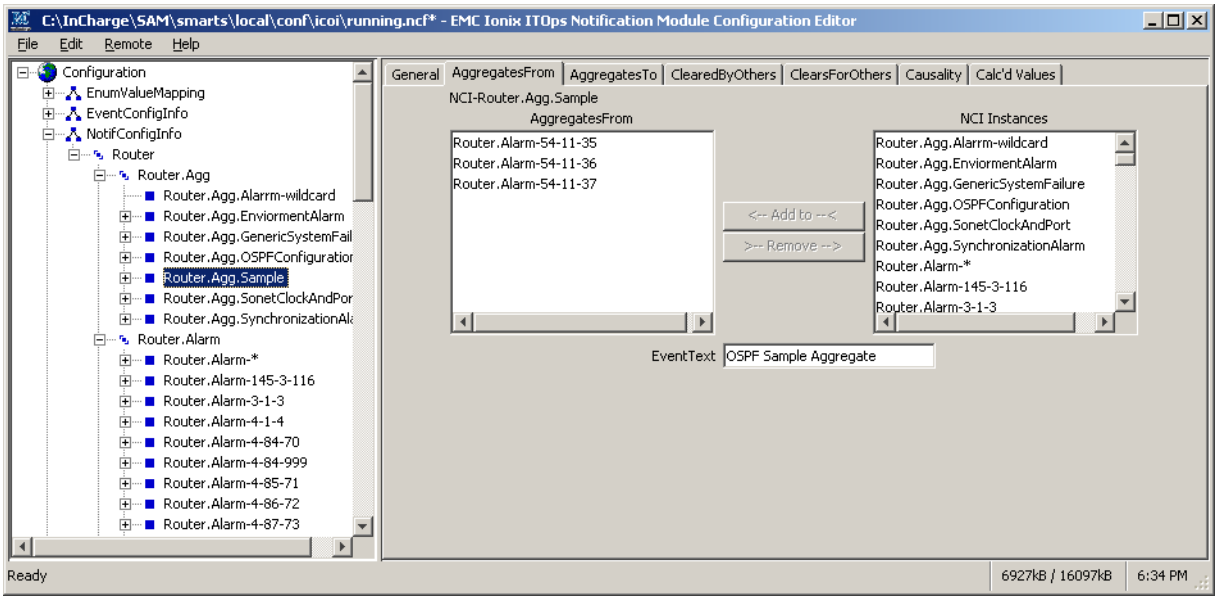


Figure 26 AggregatesFrom tab of the NCI Setup dialog box

In Figure 27 on page 61, note that unpublished notifications may aggregate to more than one aggregate notification.

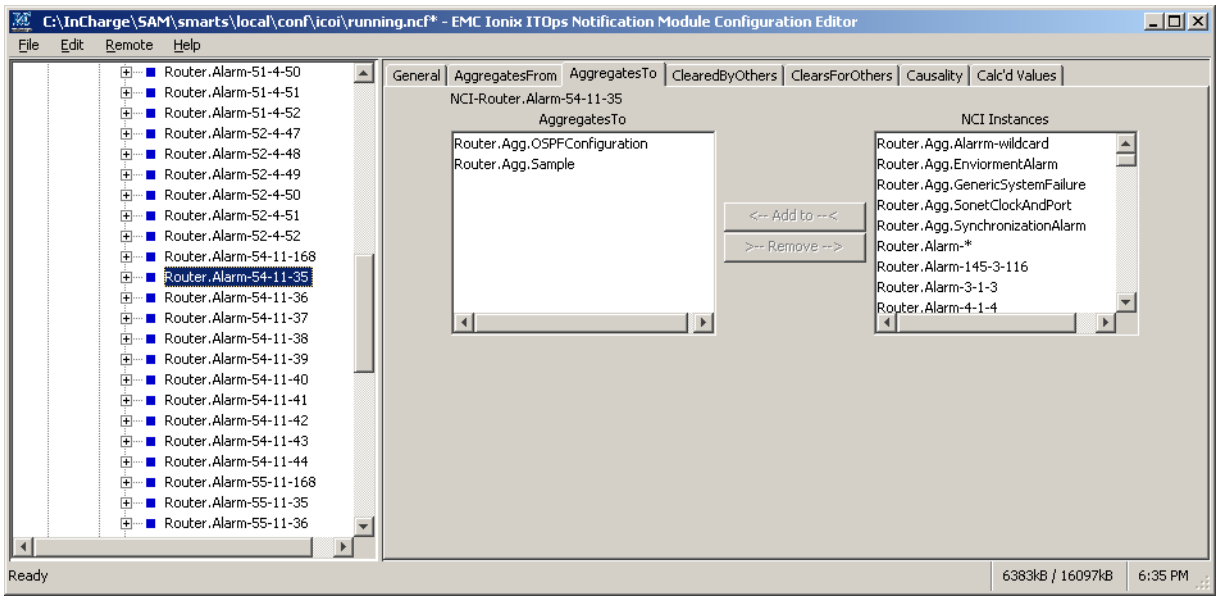


Figure 27 AggregatesTo tab of the NCI Setup dialog box

Mixing aggregation and causes

A useful feature of EMC Smarts NOTIF is the ability to relate notifications both on the aggregation level and the causes relationship level. The aggregate notification function allows you to create an independent notification and then roll other notifications under its display; the CausedBy relationship allows notifications from other servers to then be rolled under the aggregate notification. The result is that only the aggregate notification is displayed as the root cause for notifications coming from multiple servers.

NCI hook script

An NCI can be configured to call an ASL hook script just before notification publication; all other EMC Smarts NOTIF NCI processing is performed prior to this call. You can type the hook script in the ASL Hook Script field in the General tab of the NCI Setup dialog box (see Figure 18 on page 52). EMC Smarts NOTIF looks up ASL hook scripts in the local/rules or /rules directories. For example, if you type icoi-trapd/sample_hook.asl in the ASL Hook Script field in the General tab of the NCI Setup dialog box, EMC Smarts NOTIF will look up local/rules/icoi-trapd/sample_hook.asl and rules/icoi-trapd/sample_hook.asl.

NOTICE

"Writing ASL for use with EMC Smarts NOTIF NCI processing" on page 106 provides information on how to write an NCI ASL hook script.

EMC Smarts NOTIF passes the necessary information to reference the notification to the hook script. In return, the hook script passes a flag back to EMC Smarts NOTIF that indicates if the notification should be published. It is the responsibility of the hook script to determine if the notification should be deleted, modified, or related to other notifications.

The hook script is called both on setting and clearing of a notification generated by external sources, but not for an aggregate notification created by this EMC Smarts NOTIF server. In that case, the hook script is only called for the onset of the aggregate since the clearing of the aggregate is done by the internal EMC Smarts processing.

Calculated Values

The EMC Smarts NOTIF Editor allows you to define calculated value instructions for an NCI through the Calc'd Values tab. You can add and edit calculated values to an NCI in the same manner as you can for an ECI.

The only difference in defining calculated values for an NCI as compared to an ECI is that the legal assignment attributes are different. Legal assignment attributes for NCI calculated values include:

- Discard
- ElementClassName
- EventDisplayName
- EventState
- EventText
- Severity
- UserDefined1 - 20

NOTICE

"Calculated values" on page 38 provides information on adding and editing calculated values.

Copying NCI configuration parameters

You can copy NCI objects to reduce configuration time. To copy an NCI object, right-click on the NCI object and select Copy as shown in Figure 28 on page 63. The original NCI object that you copy is referred to as the *source*. The new NCI object is referred to as the *target*. The source object's properties are copied into the target object. Everything is copied except:

- The NotifConfigInfoRelationSpecifier instances that the NCI relates to
- The `CausedBy` attribute
- The `Specifies` attribute

The newly-copied NCI object therefore has the same AggregatesTo/AggregatesFrom, ClearedByOthers/ClearsForOthers relationships and general attributes, but none of the Causes/CausedBy relationships or associated NotifConfigInfoRelationSpecifiers. This is intended by design. You must configure the Causes/CausedBy relationship information manually after the copy operation.

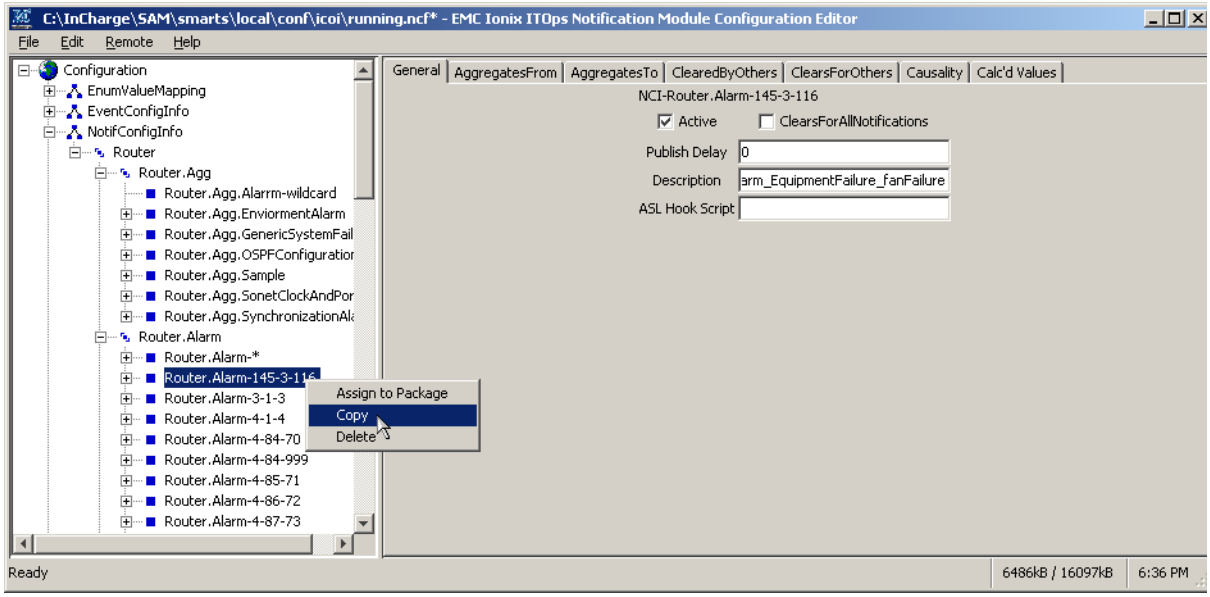


Figure 28 NCI copy feature

CHAPTER 4
Using EMC Smarts NOTIF

This chapter describes how to use EMC Smarts NOTIF in SAM or Adapter Platform servers by using the EMC Smarts NOTIF Editor.

The EMC Smarts NOTIF application is included in the EMC Smarts Service Assurance Manager server installation and the EMC Smarts NOTIF is included in the EMC Smarts Service Assurance Global Console installation. The *EMC Smarts Installation Guide for SAM, IP, ESM, MPLS, VoIP, and NPM Managers* provides information on how to install the EMC Smarts Service Assurance Manager.

This chapter consists of the following topics:

- Starting EMC Smarts NOTIF 66
- Using the EMC Smarts NOTIF Editor 70
- Generating EMC Smarts NOTIF documentation 95
- Disabling EMC Smarts NOTIF 96

Starting EMC Smarts NOTIF

When you install the EMC Smarts Service Assurance Manager version 9.2, EMC Smarts NOTIF is not enabled by default; you must start EMC Smarts NOTIF by performing the following steps:

1. Set an environment variable in the *BASEDIR* directory path to ensure that Java can be successfully started for EMC Smarts NOTIF.
   In *BASEDIR*/smarts/local/conf/runenv_cmd.sh, add the following line:

   ```
   SM_JAVA_ENABLED=YES
   ```

2. If your setup contains SNMPv3 devices, create a seedfile with the device information using the seedfile template in the IP Manager installation area.

   To use the seedfile template to create a seed file:

   a. Go to the BASEDIR/smarts/bin directory in the IP Manager installation area and type the following command to open the seedfile template:

      ```
      sm_edit conf/seedfile
      ```

   b. Press **Enter**.

   c. Add the seed entries for the systems to be discovered.

   d. Save and close the file.

3. Configure the SNMP Trap Adapter to use the Notif-trap_mgr_parse.asl script instead of the default trap_mgr_parse.asl script in order for EMC Smarts NOTIF to process SNMP traps.

   For example:

   ```
   ./sm_trapd --name=TRAP-OI --server=INCHARGE-OI --config=icoi
   --port=9013 --model=sm_actions --output
   --rules=icoi-trapd/Notif-trap_mgr_parse.asl --daemon
   ```

   NOTICE

   The **--daemon** option only works on UNIX systems, not Windows systems. Any customizations made to the original trap_mgr_parse.asl file in the BASEDIR/smarts/rules/icoi-trapd directory must be incorporated into the Notif-trap_mgr_parse.asl file in the BASEDIR/smarts/rules/icoi-trapd directory.

4. To process SNMPv3 traps, use the --seed option when starting the SNMP Trap Adapter to point to the seed file you have created, as mentioned in step 2 .

   For example:

   ```
   ./sm_trapd -n TRAP-INCHARGE-OI --server=INCHARGE-OI
   --config=icoi --port=162
   --model=sm_actions
   --seed=C:\InCharge\SAM\smarts\local\conf\icoi\seedfile
   --rules=icoi-trapd\trap_mgr_parse.asl --traceSNMP --output
   ```

5. Configure the Notif Trap Receiver as a service to automatically the SNMP Trap Adapter on system restart.

   For example:

```
sm_service install --force  --unmanaged  --startmode=runonce \
'--name=ic-notif-trap' \
'--description=Notif Trap Receiver' \
'/opt/InCharge/SAM/smarts/bin/sm_trapd' \
'--name=NOTIF-TRAP' \
'--server=INCHARGE-OI' \
'--output' \
'--config=icoi' \
'--port=1162' \
'--model=sm_actions' \
'--rules=icoi-trapd/Notif-trap_mgr_parse.asl'
```

6. Configure the Syslog Adapter to use the Notif-SysLog_mgr.asl script instead of the default SysLog_mgr.asl script in order for EMC Smarts NOTIF to process Cisco system log files.

   For example:

```
./sm_adapter.exe --NTsErViCeNaMe=ic-syslog-adapter --name=SYSLOG-OI
--server=INCHARGE-OI --tail=/var/log/syslog --model=sm_system
--model=sm_actions --output icoi-syslog/Notif-SysLog_mgr.asl
```

   NOTICE

   One or more adapters can feed a single EMC Smarts NOTIF server.

7. Configure the Notif Syslog Adapter as a service to automatically the Syslog Adapter on system restart.

   For example:

```
sm_service install --force  --unmanaged  --startmode=runonce \
'--name=ic-notif-syslog' \
'--description=Notif Syslog Adapter' \
'/opt/InCharge9/SAM/smarts/bin/sm_adapter' \
'--name=NOTIF-SYSLOG' \
'--server=INCHARGE-OI' \
'--output' \
'--config=icoi' \
'--port=1162' \
'--model=sm_actions' \
'--model=sm_system'\
'--rules=icoi-syslog/Notif-SysLog_mgr.asl'\
'--tail=/opt/InCharge9/SAM/smarts/local/logs/sample.txt'
```

8. Launch the EMC Smarts NOTIF Editor by selecting **Start** › **Programs** › **InCharge** › **EMC Smarts NOTIF Editor**.

   NOTICE

   You can also launch the editor by double-clicking the NotifGui.bat file (for Windows systems) or the NotifGui.sh file (for UNIX systems) in the *BASEDIR*/smarts/notif/editor directory.

9. Use the EMC Smarts NOTIF Editor to connect to the running Adapter Platform or SAM server. Select **Remote** › **Edit a server's settings...** in the EMC Smarts NOTIF Editor.

10. The **Connect to a Server** dialog box appears that shows the list of available Adapter Platform or SAM server connections, as shown in Figure 29 on page 68. Choose a server in the **Connect to a Server** dialog box and click **OK**.

NOTICE

"Manage connections..." on page 88 describes how to populate the list of server connections. Clicking **More...** in the **Connect to a Server** dialog box launches the **Manage Connections** dialog box where you can add additional server connections, if desired.

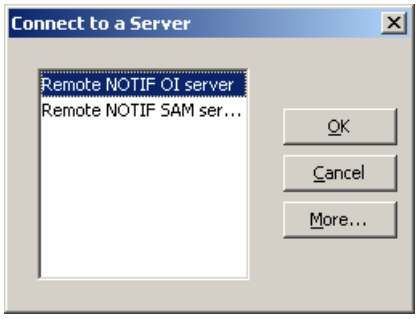


Figure 29 Connect to a Server dialog box

11. Type the password to connect to the EMC Smarts server and click **OK**.

NOTICE

The first time you connect to a given server when you use the EMC Smarts NOTIF Editor during a given work session, you will be prompted for a password. After that, you are not prompted for a password on that server for the remainder of your work session. If you restart the EMC Smarts NOTIF Editor, you must type the password again the first time you use a connection.

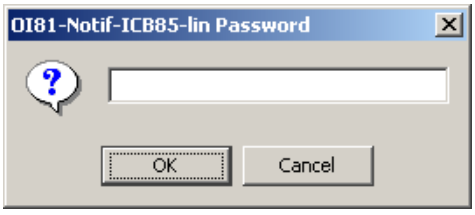


Figure 30 Server connection Password dialog box

12. The **Remote Server Settings** dialog box appears that shows the server's current settings as shown in Figure 31 on page 69. In the **Remote Server Settings** dialog box, select the **Activate EMC Smarts NOTIF** checkbox, and then click **OK**. You can also change other server settings in the **Remote Server Settings** dialog box, if desired.

NOTICE

"Edit a server's settings..." on page 87 provides more information on remote server settings.

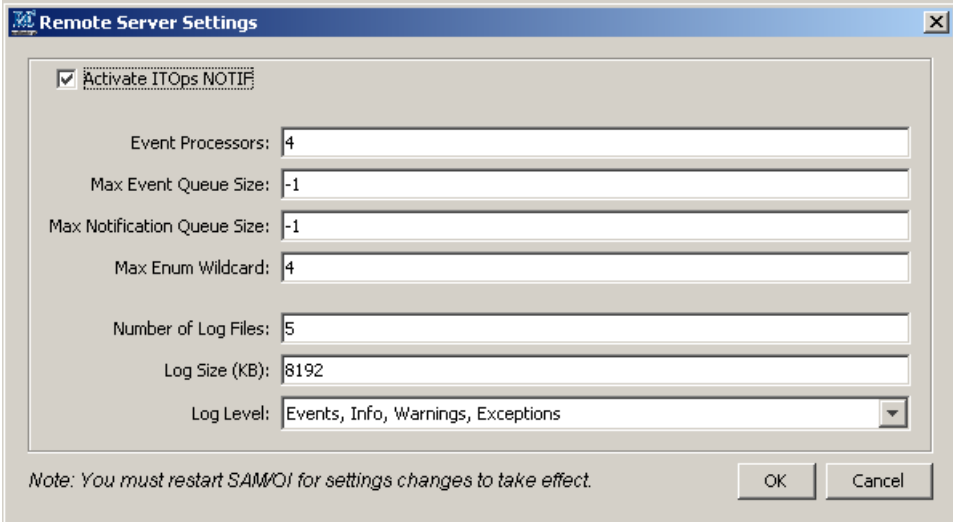


Figure 31 Remote Server Settings dialog box

13. Restart the Adapter Platform/SAM server. EMC Smarts NOTIF is now activated on the Adapter Platform/SAM server.

NOTICE

Server setting changes made in the Remote Server Settings dialog box are persistent. When you change server settings in the Remote Server Settings dialog box and click OK, the changes are saved to the Notif_Settings.import file that is generated and saved to the *BASEDIR*/smarts/local/conf/‹*icoi* or *ics*› directory in the Adapter Platform or SAM server where EMC Smarts NOTIF is running.

Server setting changes made in the EMC Smarts NOTIF Editor are preserved even if the repository is deleted. For example, if the repository was erased because you used the **--norestore** option for server startup, the server would still use the last saved settings from the EMC Smarts NOTIF Editor saved in *BASEDIR*/smarts/local/conf/‹*icoi or ics*›/Notif_Settings.import.

Using the EMC Smarts NOTIF Editor

The EMC Smarts NOTIF Editor allows you to save and restore configurations into the internal ECI and NCI configuration objects used by the underlying event management processes. Through the EMC Smarts NOTIF Editor , you can view the ECI and NCI objects and the configuration settings associated with them through a simple tree-view structure. As shown in Figure 32 on page 70, the left pane of the EMC Smarts NOTIF Editor is used to navigate through EMC Smarts NOTIF objects, and the right pane displays configuration information about the selected object.

NOTICE

The title bar for the EMC Smarts NOTIF Editor shows the source of the current configuration. In this case, it is an EMC Smarts NOTIF configuration file named running.ncf in the user's *BASEDIR*/smarts/local/conf/icoi directory.

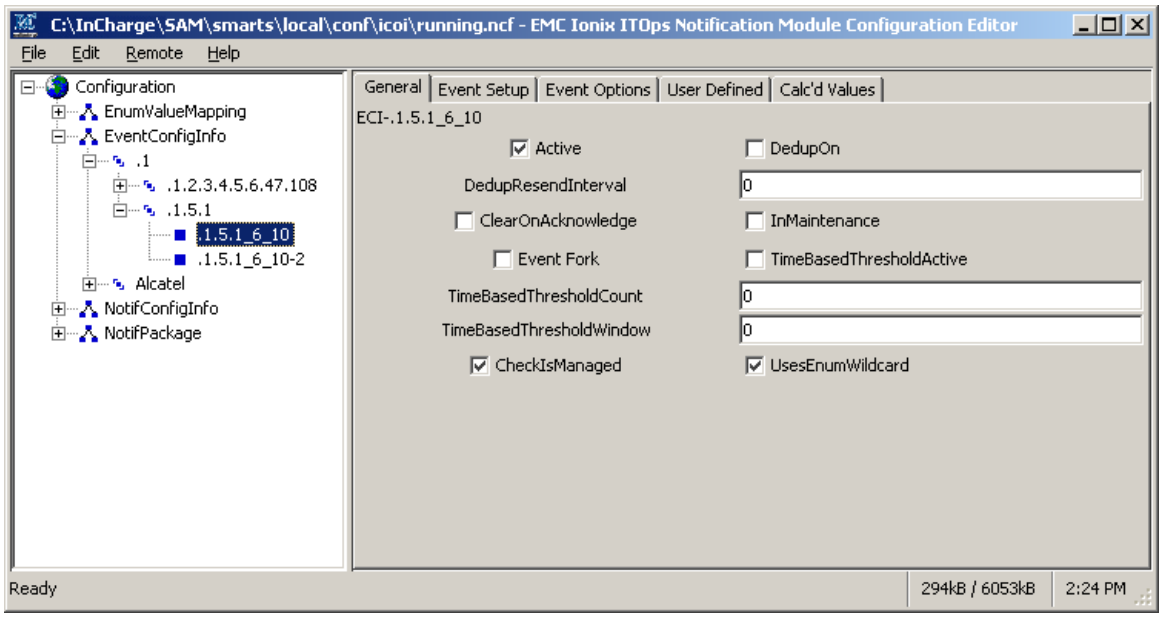


Figure 32 EMC Smarts NOTIF configuration objects in the EMC Smarts NOTIF Editor topology tree

EMC Smarts NOTIF Editor tree view

The EMC Smarts NOTIF Editor allows you to browse the configuration objects associated within the server for each of the main classes. In the EMC Smarts NOTIF Editor topology tree view, there are four main classes:

- EnumValueMapping
- EventConfigInfo
- NotifConfigInfo
- NotifPackage

The expanded EMC Smarts NOTIF Editor view in Figure 33 on page 71 shows the EnumValueMapping configuration parameters for the EVM object named `1.2.3.4.5.6.47.108.9`. An ECI may be configured such that it requires an enumeration table lookup (an EVM object) based on the values in the incoming raw event. For example,

an ECI that is matched to a raw event with a keyword | value pair (OID | value for SNMP) of `1.2.3.4.5.6.47.108.9|3` will result in a value of informational when EMC Smarts NOTIF checks the enumerated value mapping table as shown in Figure 33 on page 71.

NOTICE

"Enumeration of keyword/value pairs" on page 31 provides more information on enumerated value mapping.

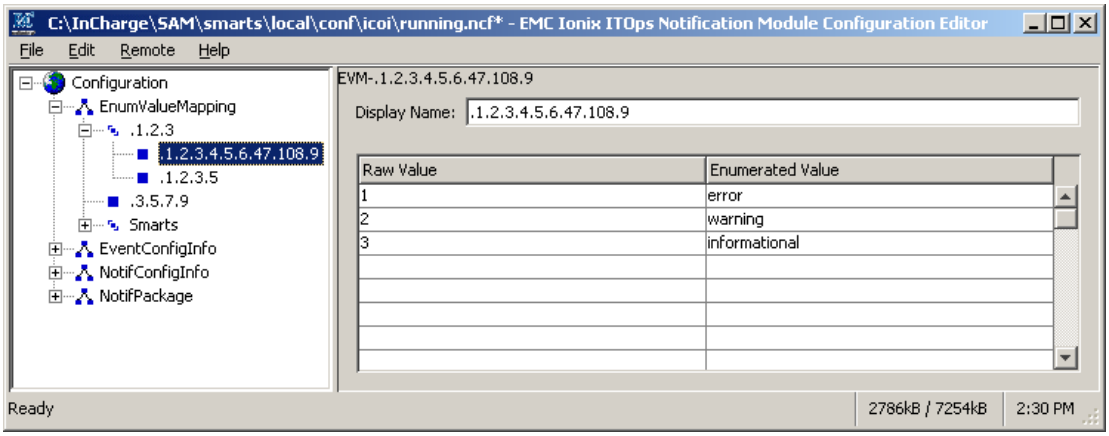


Figure 33 Enumerated value mapping for EVM-.1.3.6.1.4.1.9.9.385.1.3.1

Figure 34 on page 72 shows the expanded view of the EventConfigInfo class. The figure shows the SNMP trap configuration parameters for a specific ECI named `.1.2.3.4.5.6.47.108.10_6_10`.

NOTICE

"Setting up ECI configuration parameters" on page 32 provides more information on setting configuration parameters for ECI objects.

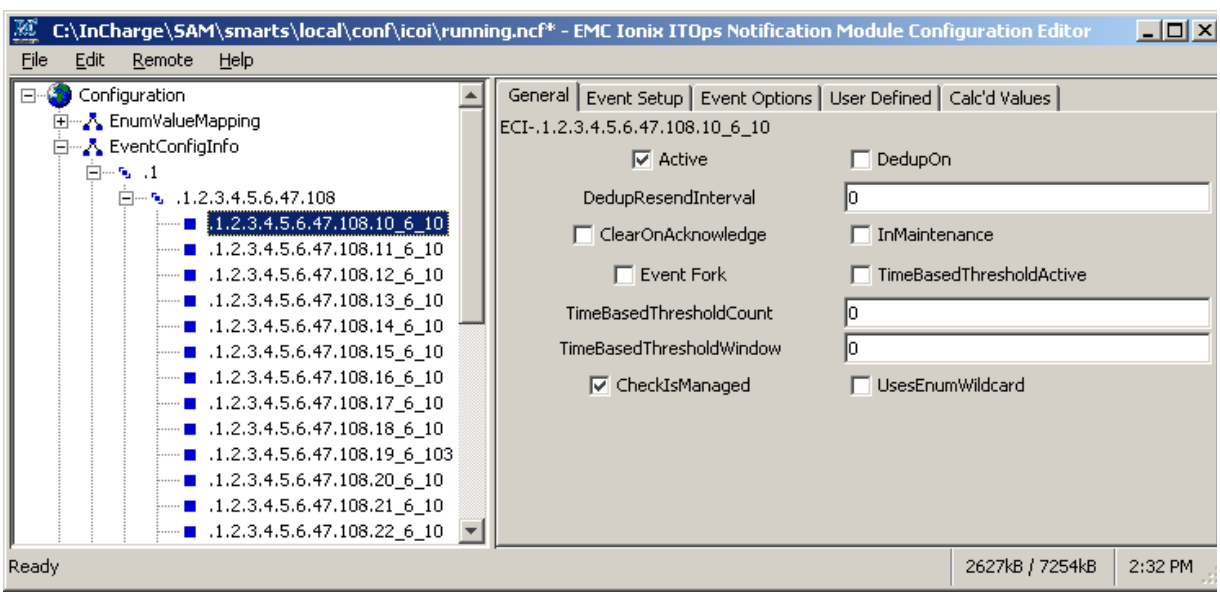


Figure 34 SNMP trap configuration parameters for the ECI named 1.2.3.4.5.6.47.108.10_6_10

Figure 35 on page 72 shows the expanded view of the NotifConfigInfo class. Note that there is a Router.Alarm grouping of NotifConfigInfo objects: all the NCIs associated with router alarms are listed in this group. Figure 35 on page 72 shows the configuration parameters for a specific NCI named `Router.Alarm-4-84-70`.

NOTICE

"Setting up NCI configuration parameters" on page 51 provides more information on setting configuration parameters for NCI objects.

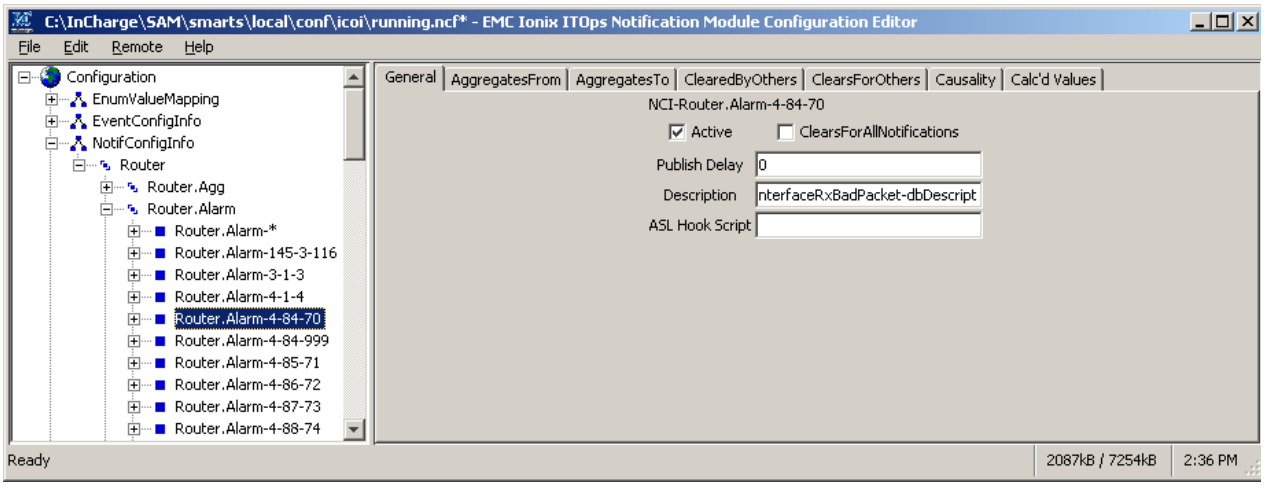


Figure 35 Configuration parameters for the NCI named Router.Alarm-4-84-70

Figure 36 on page 73 shows the expanded view of the NotifPackage class. EMC Smarts NOTIF packages allow you to group EMC Smarts NOTIF objects into logical areas of separation. For example, a set of ECIs, NCIs, and EnumValueMapping objects that support SNMP traps might be assigned to the `SNMP_Traps` package as seen in Figure 36 on page 73.

NOTICE

An object may only be assigned to a single package.

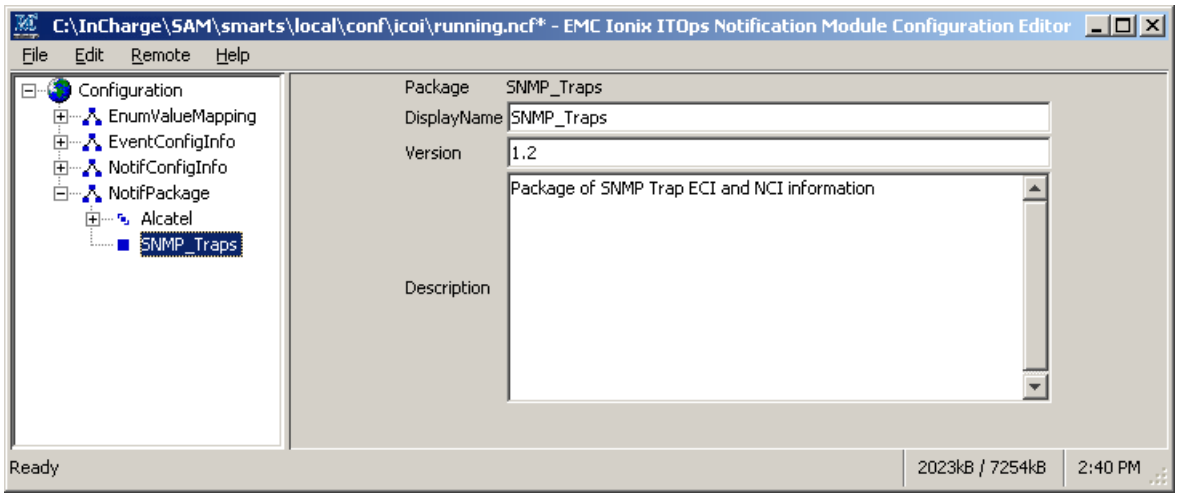


Figure 36 Settings for the NotifPackage named SNMP_Traps

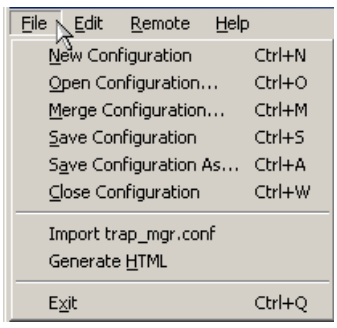
Main menu options

The EMC Smarts NOTIF Editor allows you to manipulate EMC Smarts NOTIF configuration files (*.ncf), both on the local machine that is running the EMC Smarts NOTIF Editor as well as remote servers. The EMC Smarts NOTIF Editor accesses local configuration *.ncf files under the File menu option, while remote operations are located under the Remote menu option.

The EMC Smarts NOTIF Editor provides key functions through the main menu such as: managing connections to servers, editing server settings, creating, changing, merging, and saving configurations, retrieving configurations from remote servers, storing configurations on remote servers, and generating HTML documentation. The following sections describe all of the main menu functions.

File menu options

This section describes the options available under the File menu for local *.ncf file configurations. The File menu options are described in the order they appear in the EMC Smarts NOTIF Editor, as shown below.



The File menu options associated with manipulating *.ncf files on your local server are:

- "New Configuration" on page 74
- "Open Configuration..." on page 74
- "Merge Configuration..." on page 75
- "Save Configuration" on page 76
- "Save Configuration As..." on page 76
- "Close Configuration" on page 76

The File menu options associated with importing a trap_mgr.conf file to your local server and generating an HTML version of all NCI objects are:

- "Import trap_mgr.conf" on page 76
- "Generate HTML" on page 78.

New Configuration

This option creates a new configuration on the local server (that is, a new local *.ncf configuration file). The New Configuration option erases the memory of the EMC Smarts NOTIF Editor and opens a blank topology tree in the left pane with an untitled configuration file, as shown in Figure 37 on page 74.

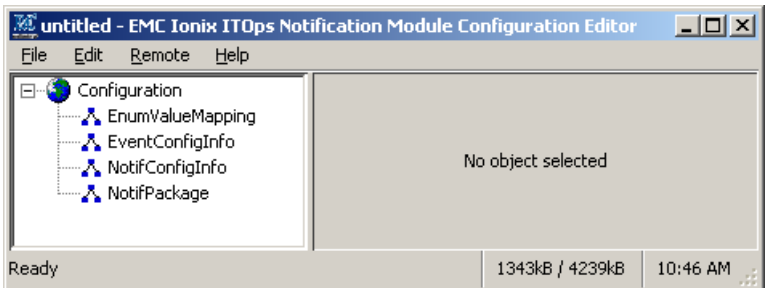


Figure 37 New Configuration window in the EMC Smarts NOTIF Editor

Open Configuration...

This option opens an existing configuration file on the local server. When this option is selected, EMC Smarts NOTIF looks for any *.ncf files in the *BASEDIR*/smarts/local/conf/‹*config*› directory or the *BASEDIR*/smarts/conf/‹*config*› directory on the local server.

When you select File › Open Configuration, the Open Notif Config File dialog box appears, as shown in Figure 38 on page 74. Select the *.ncf file you would like to open and click Open. The selected configuration is loaded into the EMC Smarts NOTIF Editor.

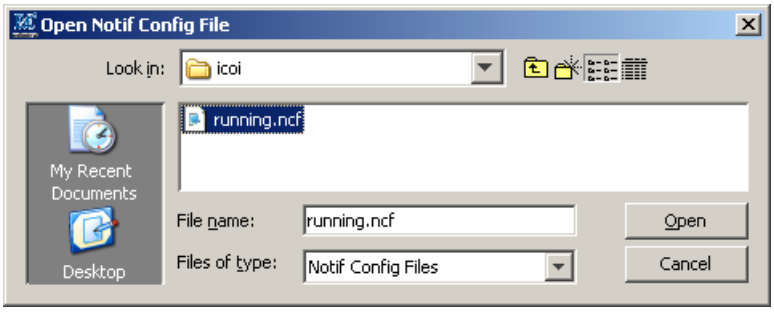


Figure 38 Open Notif Config File dialog box

Merge Configuration...

This option adds another *.ncf file into the existing configuration in memory. Unlike the Create or Open Configuration options, the Merge Configuration option adds to or overwrites the existing configuration.

When you select this option, the Merge Notif Config File dialog box appears as shown in Figure 39 on page 75. Select the *.ncf file that you want to merge with your existing configuration and click Open.

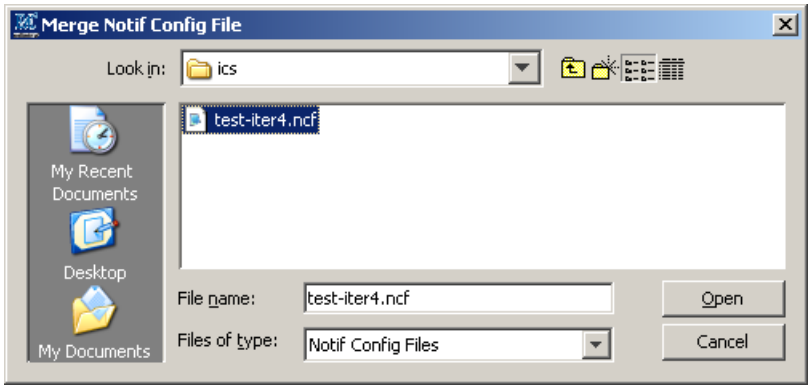


Figure 39 Merge Notif Config file dialog box

If the two configurations (that is, the existing configuration and the configuration you want to merge) have the same object, a prompt appears to choose between Keep Existing, Always Keep Existing, Use Imported, or Always Use Imported (see Figure 40 on page 75). The “Always" options stop the prompting and EMC Smarts NOTIF uses that choice for the remainder of the merge operation.

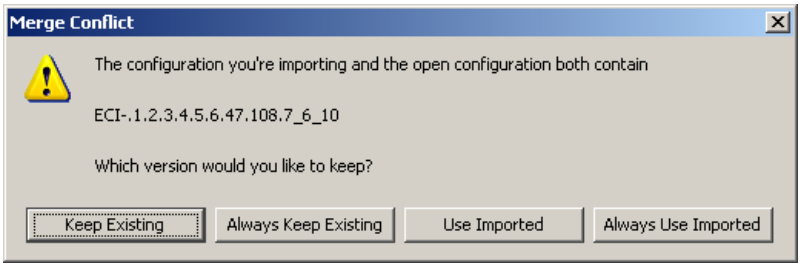


Figure 40 Merge Conflict dialog box

Save Configuration

This option saves the existing configuration to an *.ncf file on the local server. If the existing configuration is untitled, the EMC Smarts NOTIF Editor will prompt for a filename, and the Save dialog box will display, as shown in Figure 41 on page 76. Name your *.ncf file and click Save.

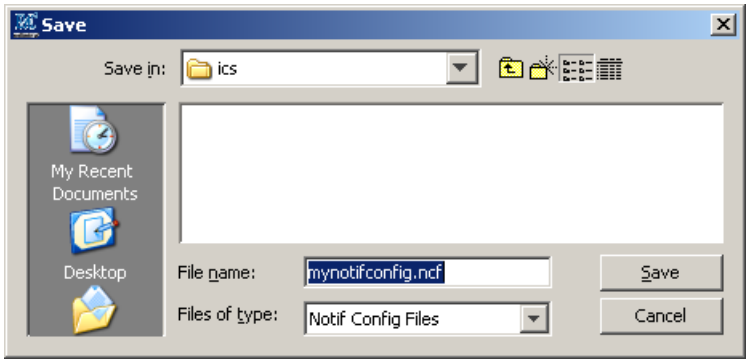


Figure 41 Save dialog box

Save Configuration As...

This option saves your existing configuration to a new *.ncf file on the local server. You can name or rename (if the existing configuration already has a name) an existing configuration to a specified filename. When you select this option, the Save dialog box appears as shown in Figure 41 on page 76. Name your *.ncf file and click Save.

Close Configuration

The Close Configuration option clears the EMC Smarts NOTIF Editor memory.

Import trap_mgr.conf

The trap_mgr.conf files used to configure a TRAP-INCHARGE-OI server (SNMP Trap Adapter) may be loaded into EMC Smarts NOTIF to replicate the standard file-based functionally. All .conf file features and options are supported. Trap_mgr.conf files can be imported into a new configuration (File › New Configuration) or an existing open configuration (File › Open Configuration...) in the EMC Smarts NOTIF Editor.

To import a trap_mgr.conf file into a new or open configuration:

1. In the EMC Smarts NOTIF Editor, select **File › Import trap_mgr.conf**.
2. The **Loading trap_mgr.conf** dialog box appears as shown in Figure 42 on page 77. Select the trap_mgr.conf file that you want to import and click **Open**.

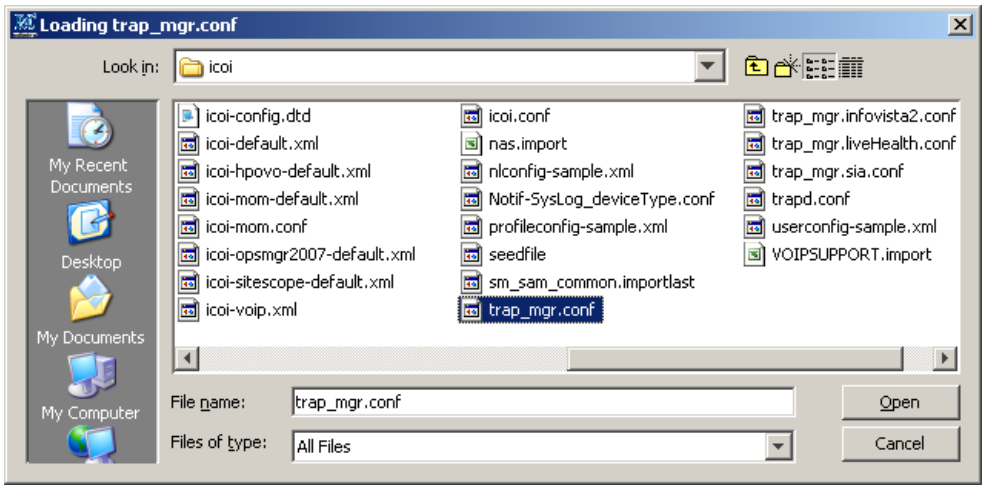


Figure 42 Loading trap_mgr.conf dialog box

The configuration from the trap_mgr.conf file is loaded into the EMC Smarts NOTIF Editor.

NOTICE

If the trap definitions contain MAP statements, they will not be imported. You need to manually specify a value for the EnumeratedValue option in order to obtain the Map statements. Mapping statement functionally, (trap/varbind position dependant), is replaced with true enumerated value substitutions, (EVM's in NOTIF), that require OID's for generic substitution. Aggregation definitions within trap_mgr.conf generally reference substitution names such as 'E-G-$SomethingElse$,' which is undetermined within the scope of NOTIF aggregation. The user should review the trap_mgr.conf file to determine what EVM's and aggregations need to be created in NOTIF given the appropriate OID's and names.

3. In the EMC Smarts NOTIF Editor, click **File › Save Configuration** or **File › Save Configuration As...** to save the imported configuration.

NOTICE

When you import configuration files, they only become persistent in the Adapter Platform or SAM server when you click **Save Configuration** or **Save Configuration As...** Otherwise, the imported configuration is only in memory (non-persistent).

Generate HTML

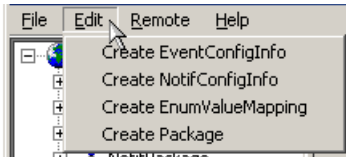
This option creates an HTML version of all NCI objects for review and documentation purposes. Index.html will provide the link to each EMC Smarts NOTIF instance (see "Generating EMC Smarts NOTIF documentation" on page 95).

Exit

This option provides clean shutdown of the EMC Smarts NOTIF Editor.

Edit menu options

This section describes the options available under the Edit menu for open configurations. The Edit menu options are described in the order they appear in the EMC Smarts NOTIF Editor, as follows:



The Edit menu options are:

- "Create EventConfigInfo" on page 78
- "Create NotifConfigInfo" on page 79
- "Create EnumValueMapping" on page 79
- "Create Package" on page 79

Create EventConfigInfo

This creates a new ECI object through the EMC Smarts NOTIF Editor. When you select this option, the dialog box shown in Figure 43 on page 78 appears.

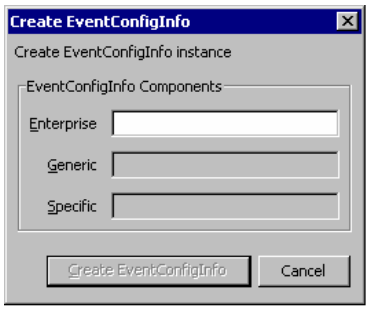


Figure 43 Create EventConfigInfo feature

Create NotifConfigInfo

This creates a new NCI object through the EMC Smarts NOTIF Editor. When you select this option, the dialog box shown in Figure 44 on page 79 appears.

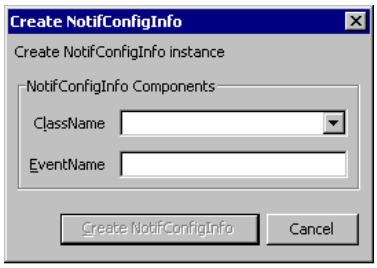


Figure 44 Create NotifConfigInfo feature

Create EnumValueMapping

This creates a new EnumValueMapping object through the EMC Smarts NOTIF Editor. When you select this option, the dialog box shown in Figure 45 on page 79 appears.

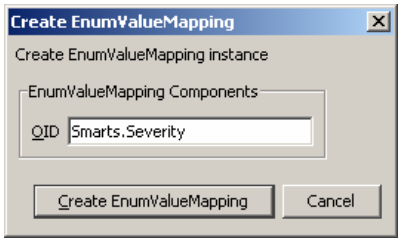


Figure 45 Create EnumValueMapping feature

Create Package

Packages can be defined to group EMC Smarts NOTIF objects into sets for maintainability and transportability. These package sets can be created as a new NotifPackage object through the Create Package option. When you select this option, the dialog box as shown in Figure 46 on page 79 appears.

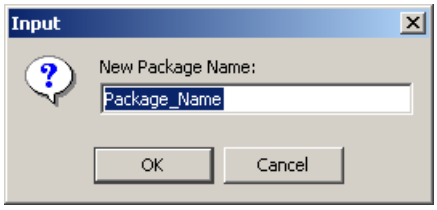
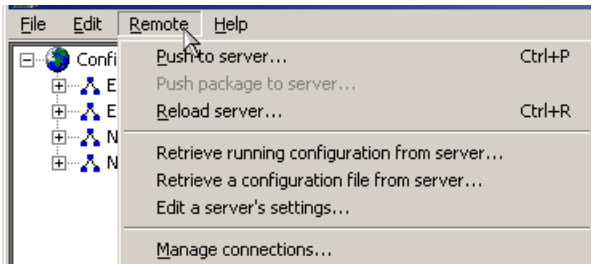


Figure 46 Create Package feature

Remote menu options

This section describes the options available under the Remote menu for remote *.ncf file configurations. The Remote menu options are described in the order they appear in the EMC Smarts NOTIF Editor, as follows:



The Remote menu options associated with pushing (or storing) a local *.ncf configuration file or package to a remote server are:

- "Push to server..." on page 81
- "Push package to server..." on page 82
- "Reload server..." on page 83

The Remote menu options associated with retrieving a running configuration or a specific *.ncf file from a remote server to a local server are:

- "Retrieve running configuration from server..." on page 84
- "Retrieve a configuration file from server..." on page 85

The Remote › Edit a server's settings menu option is used to edit server settings on a remote server. This menu option is used to activate EMC Smarts NOTIF on a remote server, as mentioned previously in the section "Starting EMC Smarts NOTIF" on page 66. Using this menu option creates persistent server settings. "Edit a server's settings..." on page 87 provides details.

The Remote › Manage connections menu option is used to manage remote server connections. "Manage connections..." on page 88 provides details.

Push to server...

The EMC Smarts NOTIF Editor allows you to manipulate the *.ncf files on a remote EMC Smarts NOTIF server by pushing (or storing) the EMC Smarts NOTIF Editor configuration to the remote server and optionally instructing EMC Smarts NOTIF within that server to reload the *.ncf files. Reloading of the *.ncf files reloads *all* *.ncf files in the corresponding *BASEDIR*/smarts/local/conf/‹*config*› directories in the remote EMC Smarts NOTIF server.

NOTICE

Ensure there are no conflicting configurations within multiple *.ncf files as the order in which these files are read is undetermined. Use only one *.ncf file on a remote server.

The Push to server option saves the existing open configuration to a file on a remote EMC Smarts NOTIF Adapter Platform or SAM server of your choice. When you select this option:

1. The **Connect to a Server** dialog box appears as shown in Figure 29 on page 68. Choose the remote server where you want to store your configuration and click **OK**. A prompt may appear to type a password to connect to the remote server as shown in Figure 30 on page 68.

   NOTICE

   The first time you connect to a given server when you use the EMC Smarts NOTIF Editor during a given work session, type a password when the prompt appears. After that, the prompt to type a password does appear on that server.

2. The **Directory** dialog box appears as shown in Figure 47 on page 81. Choose the directory location on the remote server where you want to store your configuration and click **OK**.

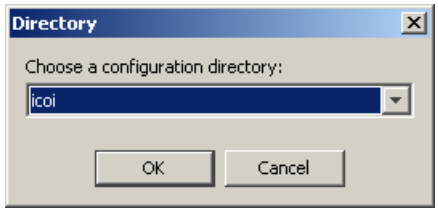


Figure 47 Directory dialog box

3. The **Select Remote File** dialog box appears as shown in Figure 48 on page 81. This is the name of the *.ncf file that you are storing on the remote server. The *.ncf file is always stored to the *BASEDIR*/smarts/local/conf/‹*config*› directory on the remote server. Select the *.ncf file you want to push to the remote server and click **OK**.

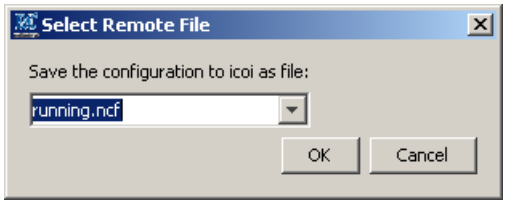


Figure 48 Select Remote File

NOTICE

If an *.ncf file already exists in the /local/conf/‹*config*› directory on the remote server, a prompt appears to overwrite the existing *.ncf file or cancel.

4. The **Push Complete** dialog box appears as shown in Figure 49 on page 82. This dialog confirms that the *.ncf file has been pushed to the remote server and prompts you to reload the EMC Smarts NOTIF processor on that server so that the new configuration takes effect.

NOTICE

Reloading of the *.ncf file on a remote EMC Smarts NOTIF server reloads *all* *.ncf files in the corresponding icoi/ics directories in that server. Care should be taken to ensure there are no conflicting configurations within multiple *.ncf files as the order in which these files are read is undetermined.

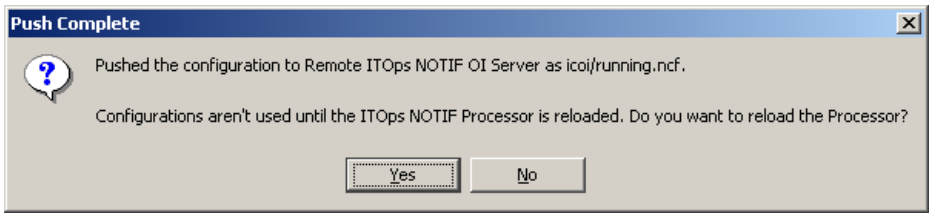


Figure 49 Push Complete dialog box

5. Click **Yes** to reload the EMC Smarts NOTIF processor to make the new configuration active on the remote server.
6. The **Reload Complete** dialog box appears as shown in Figure 50 on page 82. This dialog box confirms that the new configuration in the *.ncf file has been reloaded and is persistent in the remote server. Click **OK**.

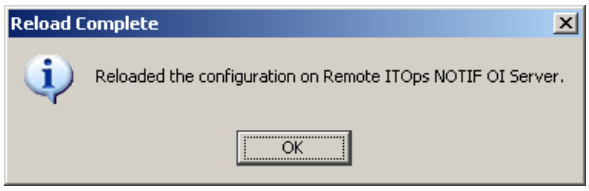


Figure 50 Reload Complete dialog box

Push package to server...

This option allows you to store a selected *.ncf package to a remote server. When you select this option:

1. The **Connect to a Server** dialog box appears as shown in Figure 29 on page 68. Choose the remote server where you want to push the package and click **OK**.
2. The **Directory** dialog box appears as shown in Figure 47 on page 81. Choose the directory location on the remote server where you want to push the package and click **OK**.

3. The **Select Remote File** dialog box appears as shown in Figure 51 on page 83. Select the name of the *.ncf package file you want to push to the remote server and click **OK**.

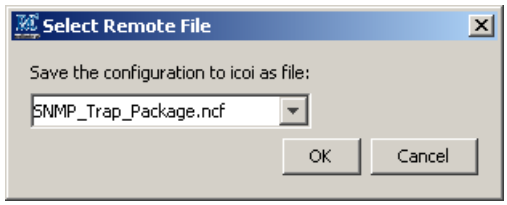


Figure 51 Select Remote File dialog box

4. The **Push Complete** dialog box appears as shown in Figure 52 on page 83. This dialog confirms that the *.ncf package file has been pushed to the remote server and prompts you to reload the EMC Smarts NOTIF processor on that server so that the new configuration package takes effect on that server.

NOTICE

Reloading of the *.ncf file on a remote EMC Smarts NOTIF server reloads all *.ncf files in the corresponding icoi/ics directories in that server. Care should be taken to ensure there are no conflicting configurations within multiple *.ncf files as the order in which these files are read is undetermined.

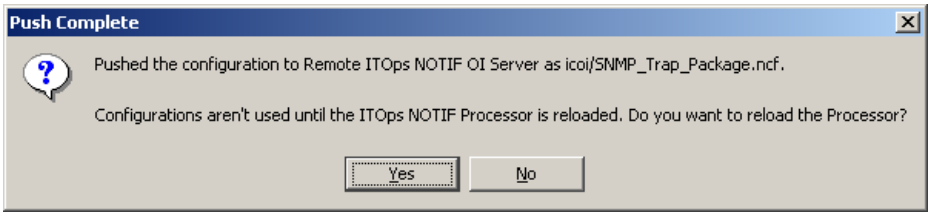


Figure 52 Push complete for package dialog box

5. Click **Yes** to reload the EMC Smarts NOTIF processor to make the new configuration package active on the remote server.
6. The **Reload Complete** dialog box appears as shown in Figure 50 on page 82. This dialog box confirms that the new configuration in the *.ncf package file has been reloaded and is persistent in the remote server. Click **OK**.

Reload server...

This option allows you to request that a remote server running EMC Smarts NOTIF reload its *.ncf configuration files. The Reload server option is used to make configuration changes active on a remote server. For example, if an *.ncf configuration file is pushed to a remote server through the Push to server option, or a specific *.ncf package file is pushed to a remote server through the Push package to server option, the configuration changes on the remote server will not become active unless the Reload server menu option is selected.

NOTICE

Care should be taken to ensure there are no conflicting configurations within multiple *.ncf files on the remote server, as the order in which these files are read is undetermined. Use only one *.ncf file on a remote server.

When you select the Reload server option in the EMC Smarts NOTIF Editor:

1. The **Connect to a Server** dialog box appears as shown in Figure 29 on page 68. Choose the remote server where you want to reload the configuration file and click **OK**.
2. If there are other *.ncf files in the remote server besides the *.ncf file you are pushing from your local server to the remote server, note that *all* of the *.ncf files on the remote EMC Smarts NOTIF Adapter Platform or SAM server will be reloaded in the corresponding *BASEDIR*/smarts/local/conf/‹*config*› directory.
3. After the remote server has reloaded the configuration file(s), the **Reload Complete** dialog box shown in Figure 50 on page 82 appears. This dialog box confirms that the new configuration in the *.ncf file has been reloaded and is persistent in the remote server. Click **OK**.

Retrieve running configuration from server...

This option retrieves a running configuration from a remote server (that is, the existing configuration running in the EMC Smarts NOTIF Editor on the remote server). When you select this option:

1. The **Connect to a Server** dialog box appears (see Figure 29 on page 68). Choose the remote server from which you want to retrieve the running configuration and click **OK**.
2. A progress bar at the bottom of the screen indicates that the running configuration from the remote server is being loaded into the EMC Smarts NOTIF Editor on the local server, as shown in Figure 53 on page 84. In this example, the running configuration being retrieved is a configuration file named running.ncf on a remote Adapter Platform server named OI81-Notif-ICB88-lin.

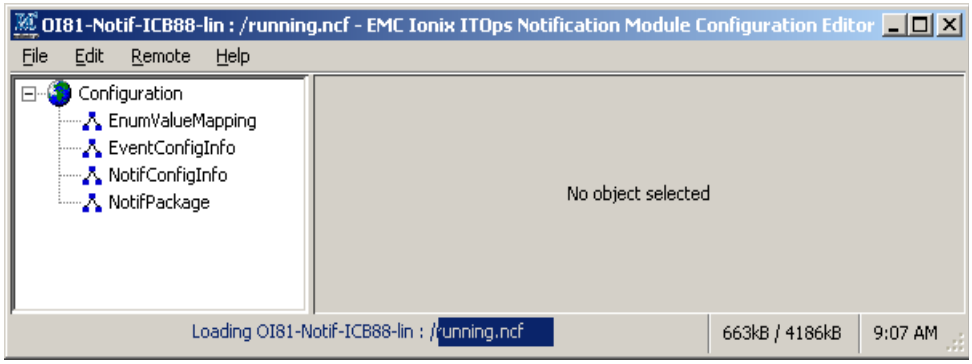


Figure 53 Loading a running configuration from a remote server

3. When the configuration data from the remote server is imported into the EMC Smarts NOTIF Editor on the local server:
   - The EMC Smarts NOTIF configuration objects in the topology tree are populated (that is, there are now EnumValueMapping, EventConfigInfo, NotifConfigInfo, and NotifPackage objects in the tree), and

- The word “Ready” appears in the bottom-left corner of the EMC Smarts NOTIF Editor, as shown in Figure 54 on page 85.

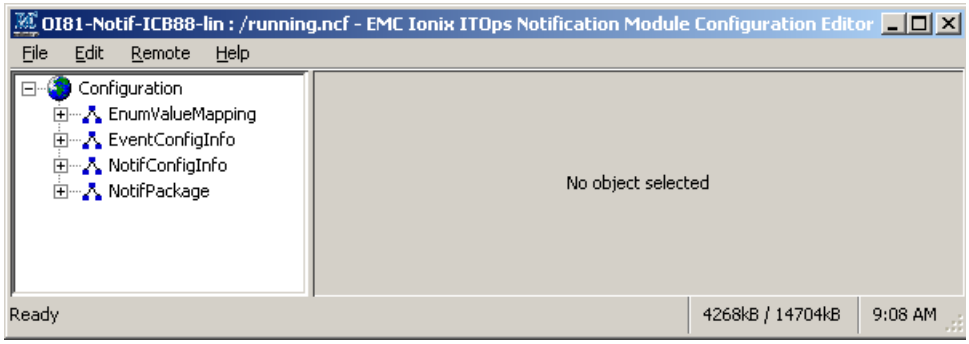


Figure 54 Running configuration from remote server loaded into EMC Smarts NOTIF Editor

4. To save the configuration to your local server:
   a. Select **File › Save Configuration As...**
   b. Name the .ncf file and select the directory on your local server where you want to store the configuration in the **Save** dialog box as shown in Figure 41 on page 76.

Retrieve a configuration file from server...

This option retrieves a specific named *.ncf file from the remote server’s configuration directories (that is, from the *BASEDIR*/smarts/conf/‹*config*› or *BASEDIR*/smarts/local/conf/‹*config*› directories). When you select this option:

1. The **Connect to a Server** dialog box appears as shown in Figure 29 on page 68. Choose the remote server from which you want to retrieve the specific *.ncf file and click **OK**.
2. The **Directory** dialog box appears as shown in Figure 47 on page 81. Choose a directory on the remote server from which you want to retrieve the *.ncf file and click **OK**.
3. The **File** dialog box appears, as shown in Figure 55 on page 85. This dialog provides a drop-down list of the *.ncf files in the selected directory.

NOTICE

If you are retrieving an *.ncf file from a remote server, and you have existing *.ncf file(s) on your local system, be careful to ensure that there are no conflicting configurations within multiple *.ncf files as the order in which these files are read is undetermined.

Select the *.ncf file you want to retrieve from the remote server and click **OK**.

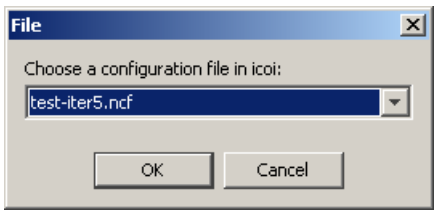


Figure 55 File dialog box

4. A progress bar at the bottom of the screen indicates that the selected remote *.ncf file is being loaded into the EMC Smarts NOTIF Editor on the local server, as shown in Figure 56 on page 86. In this example, the *.ncf file being retrieved is a configuration file named test-iter5.ncf on a remote Adapter Platform server named OI81-Notif-ICB88-lin.

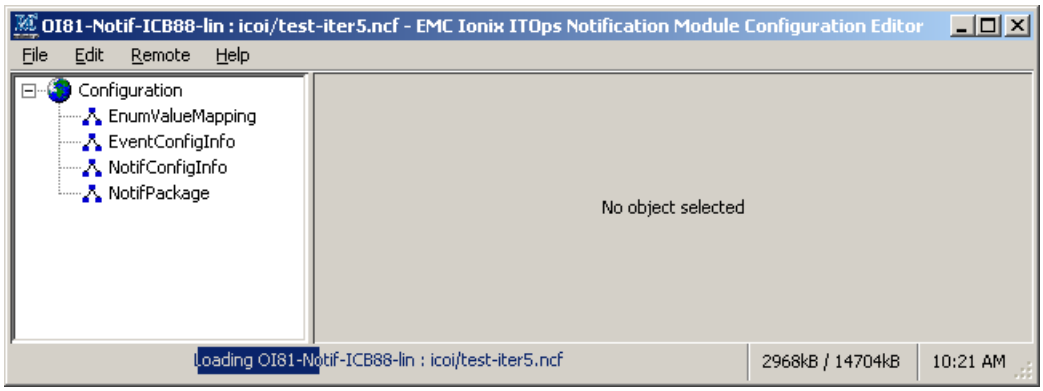


Figure 56 Loading an .ncf file from a remote server

5. When the *.ncf file from the remote server is imported into the EMC Smarts NOTIF Editor on the local server:
   - The EMC Smarts NOTIF configuration objects in the topology tree are populated (that is, there are now EnumValueMapping, EventConfigInfo, NotifConfigInfo, and NotifPackage objects in the tree), and
   - The word “Ready” appears in the bottom-left corner of the EMC Smarts NOTIF Editor as shown in Figure 57 on page 86.

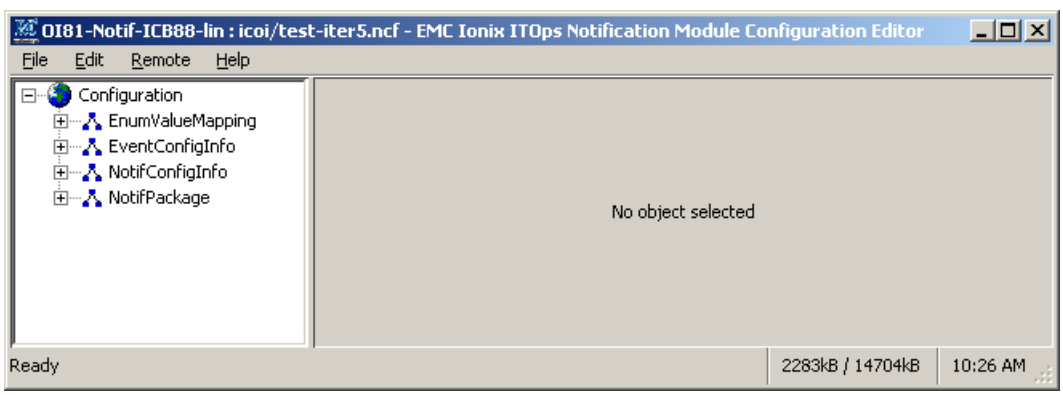


Figure 57 .ncf file from a remote server is now loaded into the EMC Smarts NOTIF Editor

6. To save the configuration to your local server:
   a. Select **File › Save Configuration As...**
   b. Name the .ncf file and select the directory on your local server where you want to store the configuration in the **Save** dialog box (see Figure 41 on page 76).

Edit a server's settings...

This option is used to change server settings on remote Adapter Platform/SAM servers. One of the settings you can change on a remote server is whether EMC Smarts NOTIF is active on that server. The process for activating EMC Smarts NOTIF on a remote server using the Edit a server's settings... option is described in "Starting EMC Smarts NOTIF" on page 66.

To change settings on a remote Adapter Platform/SAM server:

1. Select **Remote › Edit a server's settings**.
2. The **Connect to a Server** dialog box appears, as shown in Figure 29 on page 68. Choose the remote server where you want to change server settings and click **OK.**
3. The **Remote Server Settings** dialog box appears that shows the server's current settings as shown in Figure 58 on page 87. The options you can change in this dialog are defined in Table 2 on page 88.

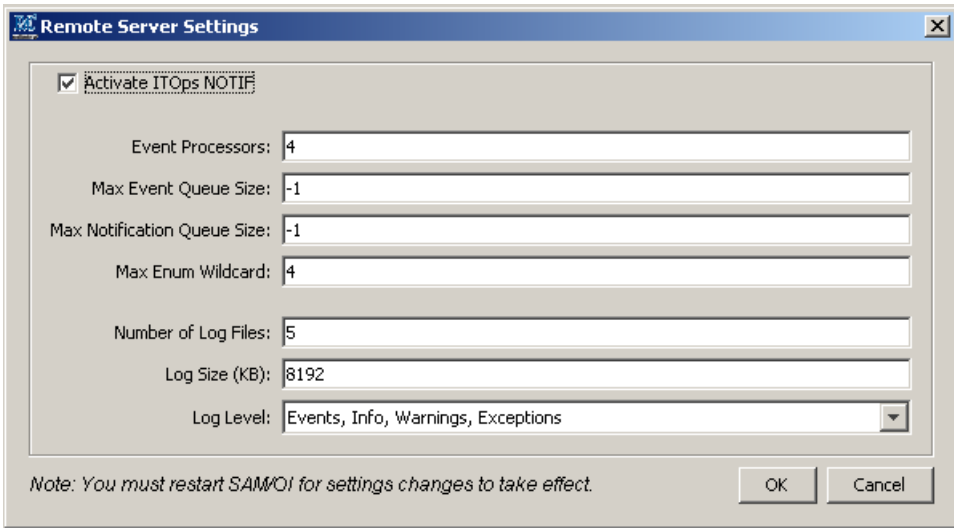


Figure 58 Remote Server Settings dialog box

4. Make any setting changes to the remote server in the **Remote Server Settings** dialog box, and click **OK**.
5. Restart the Adapter Platform/SAM server.

NOTICE

Server setting changes made in the Remote Server Settings dialog box are persistent. When you change server settings in the Remote Server Settings dialog box and click OK, the changes are saved to the Notif_Settings.import file that is generated and saved to the *BASEDIR*/smarts/local/conf/‹*icoi* or *ics*› directory in the Adapter Platform or SAM server where EMC Smarts NOTIF is running.

Table 2 Remote Adapter Platform/SAM server settings

Parameter	Type	Default value	Use
Activate EMC Smarts NOTIF	Boolean	Off	Determines whether EMC Smarts NOTIF is active on the server.
Event Processors	Integer	4	The number of threads desired to process incoming raw events. This setting also implicitly determines how many connections will exist in the EMC Smarts NOTIF connection pool. It is recommended that customers work with an EMC representative to determine the best ProcessCount number for their particular deployment.
Max Event Queue Size	Integer	-1	The maximum number of unprocessed events that can wait in the event queue. If the limit is exceeded, new events are dropped. The value “-1” means no limit.
Max Notification Queue Size	Integer	-1	The maximum number of unprocessed notifications that can wait in the notification queue. If the limit is exceeded, new notifications are dropped. The value “-1” means no limit.
MaxEnumWildcard	Integer	4	If an ECI's varbinds are checking for wildcard enumerations, EMC Smarts NOTIF drops up to this many nodes in looking for a match.
Number of Log Files	Integer	6	The maximum number of log files EMC Smarts NOTIF maintains before deleting old ones.
LogSize	Integer	2000000	When a EMC Smarts NOTIF log reaches this size (in kilobytes), the file is closed and a new one is started. The size of this setting must be between 0 and 2,097,152 KB inclusive.
LogLevel	Integer	Events, Info, Warnings, Exceptions	The log level options are: • Exceptions Only • Events, Info, Warnings, Exceptions • Everything, including debug info

Manage connections...

Connections are defined within the EMC Smarts NOTIF Editor to access remote servers. Multiple connections may be defined and given a reference name for convenience. To add a remote server connection in the EMC Smarts NOTIF Editor, perform the following steps:

1. Select **Remote › Manage Connections** in the EMC Smarts NOTIF Editor. The **Manage Connections** dialog box appears as shown in Figure 59 on page 89.

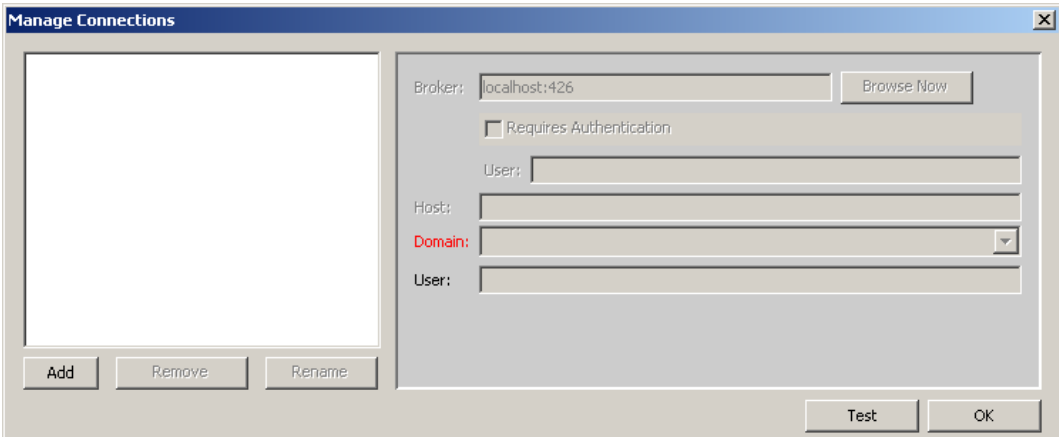


Figure 59 Blank Manage Connections dialog box

2. Click **Add**.
3. The **Input** dialog box appears as shown in Figure 60 on page 89. Type a connection reference name (for example, Remote EMC Smarts NOTIF OI server) in the **Input** dialog box and click **OK**.

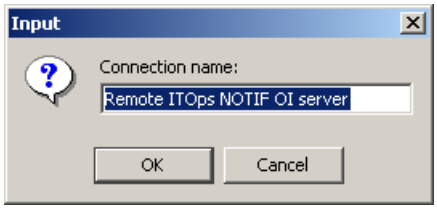


Figure 60 Input dialog box for remote server connection reference name

4. Fill in the new connection record in the right pane of the **Manage Connections** dialog box. An example is shown in Figure 61 on page 89.

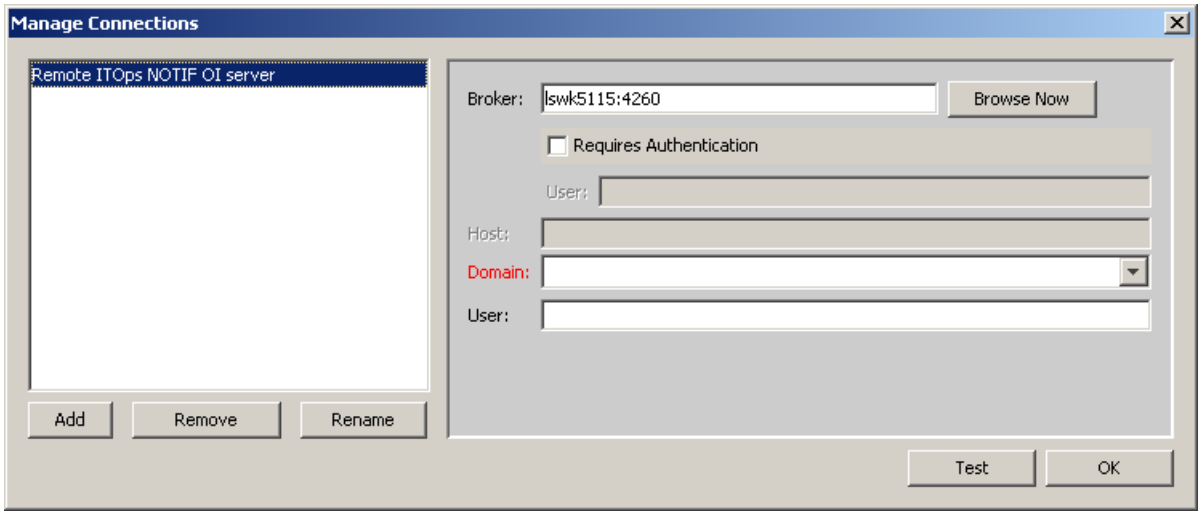


Figure 61 Manage Connections dialog box with remote server connection reference name

a. If you are connecting to a broker:

- Type the broker name in the **Broker:** field.
- Click **Browse Now** and select a domain in the **Domain:** drop-down list as shown in Figure 62 on page 90.
- Type the username in the **User:** field.

NOTICE

If you are connecting to a broker that requires authentication, select the **Requires Authentication** checkbox and type the username in the **User:** field below the checkbox.

b. If you are connecting directly to a server (that is, the **Broker:** field is empty):

- Type the hostname in the **Host:** field.
- Select a domain in the **Domain:** drop-down list.
- Type the username in the **User:** field.

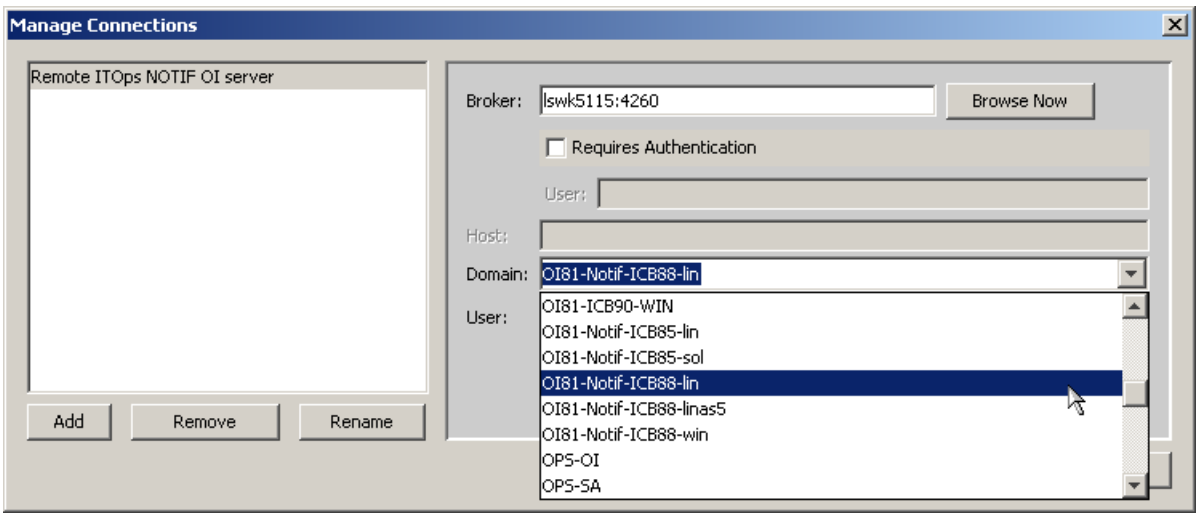


Figure 62 Selecting a domain in the Manage Connections dialog box

5. Click **Test** to try the remote server connection. If you are connected to the server, the **Connected** dialog box appears as shown in Figure 63 on page 90.

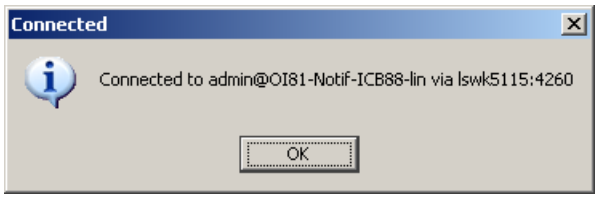


Figure 63 Connection confirmation dialog box

6. Save the connection setup by clicking **OK**. An example of a completed connection setup is shown in Figure 64 on page 91.

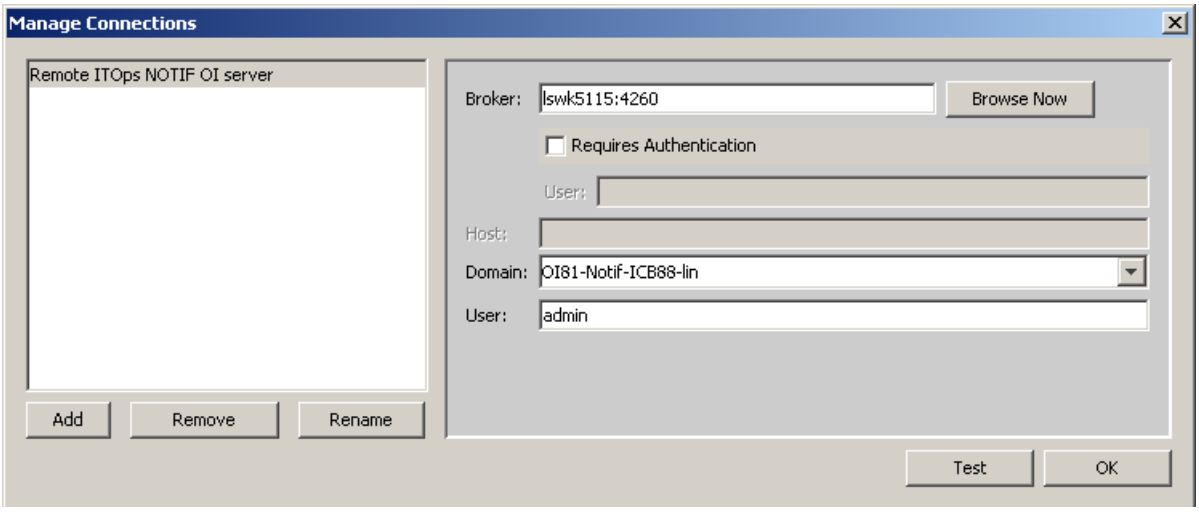


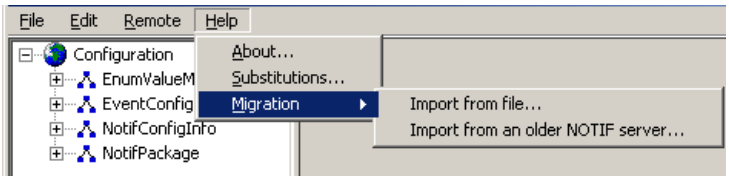
Figure 64 Completed connection setup data in the Manage Connections dialog box

NOTICE

Connections are stored in the user's preferences data on the local system, so the connection data is persistent.

Help menu options

This section describes the options available under the Help menu. The Help menu options are described in the order they appear in the EMC Smarts NOTIF Editor, as shown below.



The general Help menu options are:

- "About" on page 92
- "Substitutions" on page 92

The Help menu options associated with migrating configuration files or configurations from previous NOTIF versions to EMC Smarts NOTIF are under the Migration submenu:

- "Import from file..." on page 92
- "Import from an older NOTIF server..." on page 93

About

Provides information regarding the current version of the EMC Smarts NOTIF Editor.

Substitutions

Provides information about the substitution variables supported between raw event values and the notifications created through the EMC Smarts NOTIF Editor.

The Help menu options associated with importing configuration files to your local server are:

Import from file...

The Import from file... option loads an .import file that contains NOTIF configuration objects from a previous NOTIF version into the EMC Smarts NOTIF Editor memory. This allows you to edit the configuration (if desired) in the EMC Smarts NOTIF Editor and save the configuration into an .ncf file. The Import from file... option migrates your previous NOTIF configuration .import file into EMC Smarts NOTIF and converts it into an .ncf file format.

To migrate an import configuration file from a previous NOTIF version into the EMC Smarts NOTIF Editor memory:

1. In the EMC Smarts NOTIF Editor, select **Help › Migration › Import from file...**
2. The **Loading Instances from an Import File** dialog box appears as shown in Figure 65 on page 92. Select the .import file with the NOTIF configuration objects (from the previous NOTIF version) that you want to migrate into the EMC Smarts NOTIF Editor and click **Open**.

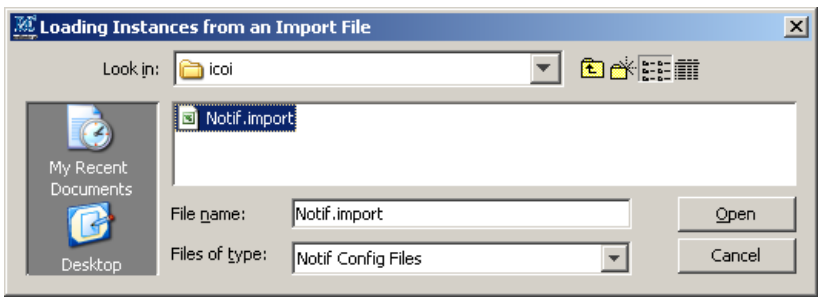


Figure 65 Loading Instances from an Import File dialog box

   The configuration from the .import file is loaded into the EMC Smarts NOTIF Editor.

3. In the EMC Smarts NOTIF Editor, click **File › Save Configuration** or **File › Save Configuration As...** to save the imported configuration.

NOTICE

When you import configuration files, they only become persistent in the Adapter Platform or SAM server when you click **Save Configuration** or **Save Configuration As...** Otherwise, the imported configuration is only in memory (non-persistent).

The Import from File option works as follows:

- If no configuration is open (that is, you just started the EMC Smarts NOTIF Editor, or you closed a configuration), the import will create a new configuration.
- If a configuration is open, the EMC Smarts NOTIF Editor will ask you if it should Cancel Import, Create New, or Add to Current (see Figure 66 on page 93).

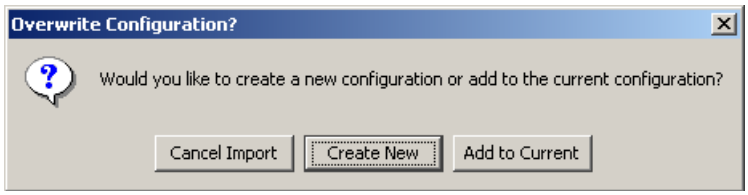


Figure 66 Overwrite Configuration dialog box

- When creating a new configuration, if the current configuration is unsaved, the EMC Smarts NOTIF Editor warns of this and asks whether to proceed, as shown in Figure 67 on page 93.

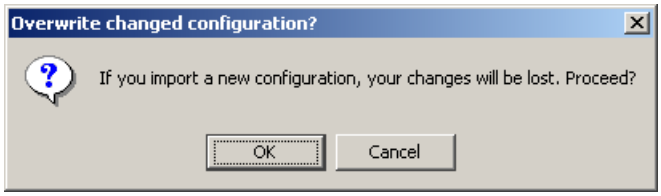


Figure 67 Overwrite changed configuration dialog box

- When reading an object (during Create New or Add to Current configuration options), if the object does not exist, it will be created with default values.
- EMC Smarts NOTIF accepts incomplete objects in the import file. Missing attributes will be default values (if the object didn't exist) or existing values.

NOTICE

If an object selected in the EMC Smarts NOTIF Editor is changed by an import operation, the EMC Smarts NOTIF Editor may not refresh the view. You have to click elsewhere and then back to the changed object to see the update.

Import from an older NOTIF server...

This option allows you to import a configuration from a running SAM 7.x/NOTIF 2.x server into the EMC Smarts NOTIF Editor and save it as an .ncf file on a SAM 9.2/EMC Smarts NOTIF server.

NOTICE

This option functions in a similar fashion as the Import from file option previously described. For example, if no configuration is open, the import will create a new configuration; if a configuration is open, the EMC Smarts NOTIF Editor will ask you if it should Cancel Import, Create New, or Add to Current, and so on.

To import a configuration from an older NOTIF server:

1. In the EMC Smarts NOTIF Editor, select **Help › Migration › Import from an older NOTIF server...**
2. The **Connect to a Server** dialog box appears, as shown in Figure 29 on page 68. Choose the running SAM 7.x/NOTIF 2.x server where you want to import the configuration from and click **OK**.

   The configuration from the running SAM 7.x/NOTIF 2.x server is loaded into into the EMC Smarts NOTIF Editor.
3. Select **Remote › Push to Server...**
4. The **Connect to a Server** dialog box appears. Choose the SAM 9.2 server where you want to push the older configuration to and click **OK**.
5. The **Directory** dialog box appears, as shown in Figure 47 on page 81. Choose the directory location on the SAM 9.2 server where you want to store your configuration and click **OK**.
6. The **Select Remote File** dialog box appears as shown in Figure 48 on page 81. This is the name of the *.ncf file you are storing on the SAM 9.2 server. The *.ncf file is always stored to the *BASEDIR*/smarts/local/conf/‹*config*› directory on the SAM 9.2 server. Name the *.ncf file that you want to push to the SAM 9.2 server and click **OK**.
7. The **Push Complete** dialog box appears, as shown in Figure 49 on page 82. This dialog confirms that the *.ncf file has been pushed to the SAM 9.2 server and prompts you to reload the EMC Smarts NOTIF processor on that server so that the new configuration takes effect.

   NOTICE

   Reloading of the *.ncf file on a remote EMC Smarts NOTIF server reloads *all* *.ncf files in the corresponding icoi/ics directories in that server. Care should be taken to ensure there are no conflicting configurations within multiple *.ncf files as the order in which these files are read is undetermined.
8. Click **Yes** to reload the EMC Smarts NOTIF processor to make the new configuration active on the SAM 9.2 server.
9. The **Reload Complete** dialog box appears as shown in Figure 50 on page 82. This dialog box confirms that the new configuration in the *.ncf file has been reloaded and is persistent in the SAM 9.2 server. Click **OK**.

NOTICE

In SAM 8.1 and above, the configurations are saved to the .ncf file and can be imported to SAM 9.2 using *Open Configuration* and *Push to server operation* . "Open Configuration..." on page 74 and "Push to server..." on page 81 provide more details.

Generating EMC Smarts NOTIF documentation

A useful function provided by EMC Smarts NOTIF through the File, Generate HTML menu option is that it automatically creates an HTML-based document that includes information on all the EMC Smarts NOTIF internal configuration objects (EVM, ECI, NCI, and NotifPackage objects). This makes it easy to view all the EMC Smarts NOTIF event and notification information in one place. As shown in Figure 68 on page 95, index.html provides a link to each EMC Smarts NOTIF instance—in the example below the description is shown for the NotifConfigInfo object named Router.Alarm-3-1-3.

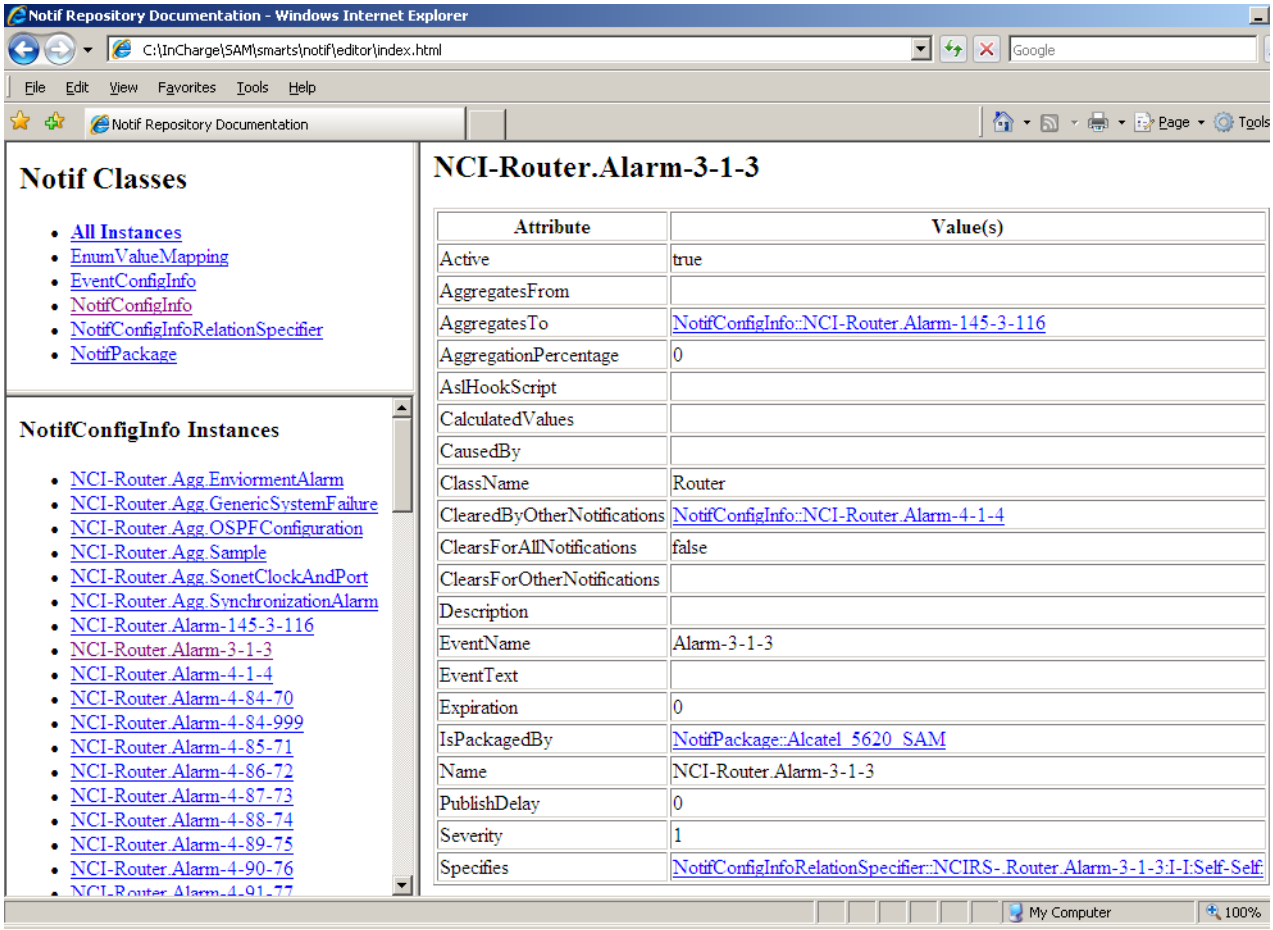


Figure 68 EMC Smarts NOTIF HTML documentation

NOTICE

Ensure you regenerate the HTML file when you modify configurations.

Disabling EMC Smarts NOTIF

The EMC Smarts NOTIF application cannot be uninstalled separately from SAM 9.2 because the EMC Smarts NOTIF processor is included as part of the SAM 9.2 server application. However, you can disable EMC Smarts NOTIF on an Adapter Platform/SAM server by performing the following steps:

1. Launch the EMC Smarts NOTIF Editor by selecting **Start** › **Programs** › **InCharge** › **EMC Smarts NOTIF Editor**.

   NOTICE

   You can also launch the editor by double-clicking the NotifGui.bat file (for Windows systems) or the NotifGui.sh file (for UNIX systems) in the *BASEDIR*/smarts/notif/editor directory.

2. Use the EMC Smarts NOTIF Editor to connect to the running Adapter Platform or SAM server where you want to disable EMC Smarts NOTIF . Select **Remote** › **Edit a server's settings...** and choose a server in the **Connect to a Server** dialog box.

3. The **Remote Server Settings** dialog box appears showing the server's current settings. In the **Remote Server Settings** dialog box, clear the **Activate EMC Smarts NOTIF** checkbox, and then click **OK**.

4. Configure the SNMP Trap Adapter to use the default trap_mgr_parse.asl script instead of the Notif-trap_mgr_parse.asl script.

   For example:

   ```
   ./sm_trapd --name=TRAP-OI --server=INCHARGE-OI --config=icoi
   --port=9013 --model=sm_actions --output
   --rules=icoi-trapd/trap_mgr_parse.asl --daemon
   ```

   NOTICE

   The **--daemon** option only works on UNIX systems, not Windows systems. Any customizations made to the Notif-trap_mgr_parse.asl file in the BASEDIR/smarts/rules/icoi-trapd directory must be incorporated into the trap_mgr_parse.asl file in the BASEDIR/smarts/rules/icoi-trapd directory.

5. Configure the Syslog Adapter to use the default SysLog_mgr.asl script instead of the Notif-SysLog_mgr.asl script.

   For example:

   ```
   ./sm_adapter.exe --NTsErViCeNaMe=ic-syslog-adapter --name=SYSLOG-OI
   --server=INCHARGE-OI --tail=/var/log/syslog --model=sm_system
   --model=sm_actions --output icoi-syslog/SysLog_mgr.asl
   ```

   NOTICE

   One or more adapters can feed a single EMC Smarts NOTIF server.

6. Restart the Adapter Platform/SAM server.

APPENDIX A
Substitution Scripting and Values

This appendix describes the scripting system used within the EMC Smarts NOTIF substitution feature and provides a list of substitution values that can be used in setting up configuration values for ECI objects.

This appendix consists of the following topics:

- Script parsing overview for substitution features 98
- Built-in operators 99
- Built-in functions 99
- Substitution values for ECI configuration parameters..... 100

Script parsing overview for substitution features

For substitution, the input is a string with embedded script. The embedded script is signalled by surrounding it with $:

- **There is no script here.**
- **There is $ *script* $ here.** — Evaluates *script* and replaces it within the string.
- **$ "There is " + *script* + " here." $** — This is equivalent. It's using + for concatenation of string literals.
- **This is a string without a $$ script** — This is an example of embedding $ in a string without starting a script block.

Scripts have identifiers, quoted strings, and numbers:

- Identifiers must start with a letter or underscore and continue until the first non-letter, non-digit, non-underscore character:
  - Examples of identifiers are **foo**, **_bar**, and **abc123**.
  - Examples of illegal identifiers are **2b** (it starts with a digit), **foo-bar** (it has an embedded hyphen).
- Quoted strings start and end with the double quote character (“). To embed a double quote, use a backslash (\) before it. To embed a backward slash, use backward slash twice (\\):
  - "This is a typical string."
  - "This is a string with \" an embedded double quote."
  - "This is a string with \\ a backslash."
  - "This is " not a legal string."

Whitespace in the script is ignored unless it is within a quoted string.

Scripts may have function calls. A function call is an identifier followed by a set of zero or more comma-separated arguments. The arguments may be any valid expression. For example:

- foo ()
- foo ("string 1", bar (identifier1, identifier2))

For example, **toUpper (E)** would report the event's enterprise value string in upper case.

NOTICE

Table 3 on page 100 lists the substitution values that can be used in ECI configuration parameters.

Built-in operators

EMC Smarts NOTIF includes the following operators for use in scripting:

- str1 + str2 — Returns the concatenation of str1 and str2.

  "abc" + "def" ---> "abcdef"

- number1 + number2 — Returns the sum of the two numbers.

  123 + 456 ---> "579"

  NOTICE

  The script evaluation of EMC Smarts NOTIF is string-based. EMC Smarts NOTIF examines the two operands and, if both are legal numbers, it sums them. Otherwise, the two strings are concatenated. To ensure a concatenation regardless of string content, use the concat() function described.

- number1 * number2 — Returns the product of the two numbers.

  3 * 4 ---> "12"

Built-in functions

EMC Smarts NOTIF includes the following built-in functions for use in scripting:

- **toLower** (string) — Returns the string in lower case.
- **toUpper** (string) — Returns the string in upper case.
- **sizeOf** (string) — Returns the length of a string.
- **substr** (string, startPosition, length) — Returns a substring of the string, starting at 'startPosition', for 'length' characters. The first character in the string is start position 1 (this is different from many programming languages that use 0 for the first position). If the length requested exceeds the length available, all of the remaining characters are returned. If the start position is less than 1, an evaluation error occurs. If the start position is beyond the length of the string, an empty string is returned.
- **concat** (string1, string2, ...) — Returns the concatenation of two or more strings. **concat ("a", "b", "c") ---> "abc"**
- **time** (), time (formatString) — Returns the current time, optionally formatted according to a format string. The format string is the typical Java language date format string (for more information, see http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html. If no format string is provided, the system uses *EEE, d MMM yyyy HH:mm:ss Z*, which results in a timestamp such as: "**Thu, 23 Aug 2007 16:01:21 -0400**."

- **extract** (string, separator, field) — Splits the string using the 'separator' and returns the 'field'. The separator value is a regular expression (for information on how to format 'separator', see http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html#split(java.lang.String). If there are not enough resulting fields, an empty string is returned:
  - extract ("1,2,3,4,5,6", "," ,4) ---› 4
  - extract ("abcdef", "c", 2) ---› "def"
  - extract ("abcdef", "c", 3) ---› ""

Substitution values for ECI configuration parameters

Table 3 on page 100 lists the substitution values that can be used in ECI configuration parameters, such as User Defined values, Event Setup options, and Calculated values.

Table 3 Substitution values for ECI configuration parameters (page 1 of 2)

Substitution Value	Description
A	The source text in the event.
E	The enterprise text in the event.
N	The ‘generic’ field in the event.
S	The ‘specific’ field in the event.
T	The timestamp in the event.
SYS	Given the source IP of the event, the node hosting the IP will be substituted.
$EVENTTEXT$	The current value of EventText.
$V1$ ----› $V20$	Replaced with the text value in the respective varbind.
$OID1$ ----› $OID20$	Replaced with the OID value in the respective varbind.
$UserDefined1$ ----› $UserDefined20$	Notification property that can be populated with any string.
$#$	Replaced with the number of varbinds in the event.
$*$	Replaced with text consisting of all varbind OIDs and values in the event.
$V*$	Replaced with text consisting of all varbind values in the event.
$V*x(n)c*$	Varbind subfield extractor. If a varbind is itself a string with subfields that can be parsed with a character token, you can extract a particular subfield by using V*x(n)c*, where: • *x* is the varbind number (*x* value represents a varbind 1–10). • *n* is the subfield desired (*n* value is a number specifying which of the subfields should be used in the substitution. Subfield numbering starts with 0). • *c* is the tokenizer character. The *c* value is a single character that can be used as a tokenizer to parse the varbind string. In many cases this is a comma. Example: if $V1 is `Tempe,AZ,85284` then $V1(2), would give `AZ`. This can also be done with the extract function. The extract function takes three parameters: string, separator, and field. Example: if $V1 is `Tempe,AZ,85284` then extract($V1, ",", 2) would give `AZ`.

Table 3 Substitution values for ECI configuration parameters (page 2 of 2)

Substitution Value	Description
$concat(string1, string2, ...)$	Will return the concatenation of two or more strings. Example: $concat(a, b, c) would return `abc`.
$contains(*textToSearch*, *textToLookFor*)$	Will return true if textToLookFor exists in textToSearch string.
$endsWith(*textToSearch*, *textToLookFor*)$	Will return true if textToLookFor is at the end of textToSearch.
$extract(*string, separator, field*)$	Will split the string using the *separator* and returns the *field*. Examples: extract ("1,2,3,4,5,6", "," ,4) ---> 4 extract ("abcdef", "c", 2) ---> "def"
$getAttribute(*className*, *instanceName*, *attributeName*)$	Will return the value of the given attribute from the given class and instance.
$sizeOf(text)$	Will return the number of characters in text. Text can itself be a substitution, such as $V1$. Example: $sizeOf(True) would produce `4`.
$startsWith(*textToSearch*, *textToLookFor*)$	Will return true if textToLookFor is at the start of textToSearch.
$substr(*text, pos, length*)$	Will extract characters from text starting at position *pos* for the specified length. Text can itself be a substitution, such as $V1$. Example: $substr(unknown,4,2) will produce `no`.
$time(*format*)$	Will produce a string based on the current time. The substitutions are as follows: • HH — 24 hour of day • MM — Minute of hour • SS — Second of minute • MTH — Month of year (numeric) • DD — Day of month • YY — Year • DOW — Day of week (3 char alphabetic) The format characters take elements from the current time to create the text. Examples: $time(*DOW HH:MM*) could produce the text `Wed 22:04`. $time(*HH:MM:SS*) could produce `15:23:40`.
$toLower(text)$	Will convert the characters in text to lower case. Text can itself be a substitution, such as $V1$. Example: $toLower(TrUe) would produce `true`.
$toUpper(text)$	Will convert the characters in text to upper case. Text can itself be a substitution, such as $V1$. Example: $toUpper(cat) would produce `CAT`.
$varbindCount$	Will return the number of varbinds.
$varbindKeyValue$	Will return a list of each varbind in "key: value" format, separated by commas.
$varbindValues$	Will return a list of varbind values, separated by commas.

APPENDIX B
ASL and Java Interfacing with EMC Smarts NOTIF

This appendix discusses how to write ASL hook scripts and Java classes for use with EMC Smarts NOTIF event processing.

This appendix consists of the following topics:

- Writing ASL for use with EMC Smarts NOTIF ECI processing 104
- Writing ASL for use with EMC Smarts NOTIF NCI processing..... 106
- Writing a Java class for use with EMC Smarts NOTIF event processing 107

Writing ASL for use with EMC Smarts NOTIF ECI processing

ASL hook scripts called from EMC Smarts NOTIF ECI specifications must include the following two additional definitions in the `Input/Output Variables` section:

```
DISCARD_TRAP = "";
UPDATE_USERDEF_FIELDS = "";
```

Sample ECI ASL hook script

A sample ECI ASL hook script is provided below for reference:

```
/*
 * Sample-ECI.asl - Hook adapter for any ECI related customizations.
 */

debug = FALSE;

// Input/Output variables.

CLASSNAME = "";
INSTANCENAME = "";
EVENTNAME = "";
SEVERITY = "";
EVENTTEXT = "";
CATEGORY = "";
EXPIRATION = "";
STATE = "";
INMAINTENANCE = "";
CLEARONACKNOWLEDGE = "";
EVENTTYPE = "";
ELEMENTCLASSNAME = "";
EVENTDISPLAYNAME = "";
ELEMENTNAME = "";
USERDEFINED1 = "";
USERDEFINED2 = "";
USERDEFINED3 = "";
USERDEFINED4 = "";
USERDEFINED5 = "";
USERDEFINED6 = "";
USERDEFINED7 = "";
USERDEFINED8 = "";
USERDEFINED9 = "";
USERDEFINED10 = "";
USERDEFINED11 = "";
USERDEFINED12 = "";
USERDEFINED13 = "";
USERDEFINED14 = "";
USERDEFINED15 = "";
USERDEFINED16 = "";
USERDEFINED17 = "";
USERDEFINED18 = "";
USERDEFINED19 = "";
USERDEFINED20 = "";

// Input Variables

TIMESTAMP = "0";
IPADDRESS = "";
SYSNAMEORADDR = "";
ENTERPRISE = "";
GENERIC = "9999";
SPECIFIC = "9999";
```

```
// Varbind and OID values
V1 = "";
V2 = "";
V3 = "";
V4 = "";
V5 = "";
V6 = "";
V7 = "";
V8 = "";
V9 = "";
V10 = "";
V11 = "";
V12 = "";
V13 = "";
V14 = "";
V15 = "";
V16 = "";
V17 = "";
V18 = "";
V19 = "";
V20 = "";

OID1 = "";
OID2 = "";
OID3 = "";
OID4 = "";
OID5 = "";
OID6 = "";
OID7 = "";
OID8 = "";
OID9 = "";
OID10 = "";
OID11 = "";
OID12 = "";
OID13 = "";
OID14 = "";
OID15 = "";
OID16 = "";
OID17 = "";
OID18 = "";
OID19 = "";
OID20 = "";

// Flag to discard the trap, when set to TRUE.
DISCARD_TRAP = "FALSE";

NotificationName = "";
MessageType = "";
NotifIntegration = "TRUE";

// Flag to update the UserDefined[1-20] fields and EventText, upon
   CLEAR
// when set to TRUE.
UPDATE_USERDEF_FIELDS = "FALSE";

if (debug) { print("Running my ECI Hook.asl "); }

START {
    .. eol
} do {

    // Default output is the same as input.

   // Set attributes in ICS_Notification.  Add any more customizations
   here ...
```

```
    USERDEFINED3 = "Added this in custom ruleset";

    if (debug) { print("Done with my ECI Hook.asl ");}
    return;
}


/*
 * These variables describe the formatting of this file.  If you don't
    * like the template defaults, feel free to change them here (not in
    * your .emacs file).

 *
 * Local Variables:
 * mode: C++
 * End:
 */
```

Writing ASL for use with EMC Smarts NOTIF NCI processing

A sample NCI ASL hook script is provided below for reference:

```
/* sampleNciHook.asl
 */
default debug = TRUE;
default detailedDebug = TRUE;

default NotificationName = "";
default DISCARD_IN_NOTIF = "";
DISCARD_IN_NOTIF = "FALSE";

icsNotificationFactory = object();
ics_domain_name = getServerName();

user = "HOOK";
aslName = this->ReadsRulesFrom->fileName;
myThread = thread();
auditText = " ".aslName." Thread=".myThread." ";


START {
    local errorOrSkip = FALSE;
    .. eol

    errorOrSkip : VERIFY_INPUT
    (!errorOrSkip)

    errorOrSkip : PRINT_SOMETHING
    (!errorOrSkip)

}


//Grab the ICS_Notification object using the name passed in.
VERIFY_INPUT {
    local error = FALSE;

    do {
        nObj = self->object(NotificationName);
        if (NotificationName == "" || nObj->isNull()) {
            print(time().auditText." Internal Error: ".
                  " could not find the notification".
```

```
                    " object('".NotificationName."')!");
            error = TRUE;
        }

        return error;
    }
}

/* Sample activity.
*/
PRINT_SOMETHING {
    local skip = FALSE;

    do {
      print(time().auditText.nObj->InstanceName." FROM NOTIF NCI
   HOOK");
    }
}
```

Writing a Java class for use with EMC Smarts NOTIF event processing

In EMC Smarts NOTIF, you can write Java classes which can be called during event processing to:

- Modify the various parameters that are set prior to the notification being created.

  or

- Instruct EMC Smarts NOTIF to discard the event it is processing.

To use a custom Java class in the EMC Smarts NOTIF processor:

1. Write and compile one or more Java classes that implement the notifEventProcessor interface with the following requirements:
   a. Your class must have a default constructor (one that takes no arguments).
   b. The EMC Smarts NOTIF .jar file must be available to your Java compiler when building your class. The .jar file is *BASEDIR*/smarts/classes/notif.jar.
2. Package your class or classes in a .jar file.
3. Install your .jar file into the *BASEDIR*/smarts/local/classes directory.
4. Update the Notif-init.asl script to include your .jar file in the CLASSPATH:
   a. Locate the Notif-init.asl file in the *BASEDIR*/smarts/rules/ics directory.
   b. Use the **sm_edit** utility to edit the script (or manually copy it to the *BASEDIR*/smarts/local/rules/ics directory and edit it there).

   NOTICE

   The *EMC Smarts System Administration Guide* provides information on how to use the **sm_edit** utility.

   c. Locate the lines that use `add_archive` to add the .jar file to the CLASSPATH.

d. Duplicate one of those lines and replace the name of the .jar file with your own .jar file. Do not specify a path for the .jar file. The EMC Smarts NOTIF/SAM Java class loader is designed to look in local/classes for your .jar file.

e. Save the file.

f. Restart the SAM/Adapter Platform server to pick up the changed CLASSPATH.

5. Add your class name to an ECI in the EMC Smarts NOTIF Editor (the **Java** field in the **Event Options** tab as shown in Figure 9 on page 36) and save and deploy the new .ncf file.

EMC Smarts NOTIF Java API

This section provides the sources for the two interfaces that comprise the EMC Smarts NOTIF Java API.

notifEventProcessor

```
package com.emc.smarts.notif.api;
public interface notifEventProcessor
{
   void process ( notifEventDataAccessor eda );
}
```

notifEventDataAccessor

```
package com.emc.smarts.notif.api;
public interface notifEventDataAccessor
{
   public String getClassName();
   public void setClassName(String className);

   public String getInstanceName();
   public void setInstanceName(String val);

   public String getEventName();
   public void setEventName(String val);

   public String getSeverity();
   public void setSeverity(String val);

   public String getEventText();
   public void setEventText(String val);

   public String getCategory();
   public void setCategory(String val);

   public String getExpiration();
   public void setExpiration(String val);

   public String getEventState();
   public void setEventState(String val);

   public boolean getInMaintenance();
   public void setInMaintenance(boolean val);

   public boolean getClearOnAcknowledge();
   public void setClearOnAcknowledge(boolean val);

   public String getEventType();
   public void setEventType(String val);

   public String getElementClassName();
   public void setElementClassName(String val);
```

```
   public String getElementName();
   public void setElementName(String val);

   public String getUserDefined ( int i ); // 1-20
   public void setUserDefined ( int i, String val );

   public boolean getDiscardTrap();
   public void setDiscardTrap(boolean val);

   // trap data
   public String getTimestamp();
   public String getSource();
   public String getSysNameOrAddr();
   public String getEnterprise();
   public String getGeneric();
   public String getSpecific();
   public String getVarbindValue(int which);
   public String getVarbindOID(int which);
}
```

An example plug-in processor

A plug-in processor example is provided below for reference:

```
package com.example.notif;

import com.emc.smarts.notif.api.notifEventDataAccessor;
import com.emc.smarts.notif.api.notifEventProcessor;

public class MyNotifPlugin implements notifEventProcessor
{
   // process gets called for each event matched to the ECI
   public void process ( notifEventDataAccessor eda )
   {
      // if VarBind1 (aka V1) has value 'DiscardMe',
      // discard this event
      final String vb1 = eda.getVarbindValue ( 1 );
      if ( vb1.equals ( "DiscardMe" ) )
      {
         eda.setDiscardTrap ( true );
      }
   }
}
```

Additional notes

- Note that varbind values are referred to by their natural numbers. The index of the value of the first varbind is 1, not 0. The example code above retrieves the value of the first varbind with `eda.getVarbindValue ( 1 );`
- When specifying your class in an ECI, use the full class name. For example, "com.example.notif.MyNotifPlugin". The string you specify is passed without modification to the JVM's default classloader via Class.forName ().
- Plug-in processors are intended to be lightweight and you must be careful not to deploy processors that do significant work in the constructor or during the call to *process*. Deploying a slow processor, especially if it is called from a frequently used ECI, can degrade performance of the entire SAM/Adapter Platform system.

- It is also important to remember that the processor class will be instantiated for each event it handles. Processors that need to retain information across calls to process() can use static member data to do so, but keeping references to the notifEventDataAccessor instance is strongly discouraged. Doing so can delay the JVM's garbage collection of internal EMC Smarts NOTIF data structures and cause the system to use more memory than usual.
- Any exceptions (including Throwable) that occur during the execution of your class are logged as warnings in the EMC Smarts NOTIF log.

INDEX

A

Active checkbox 32, 46, 47
Adapter Platform (OI) 14
add
    calculated values 39
    causes 54
AggregatesFrom 60
AggregatesTo 60, 61
aggregation 60, 61
ASL
    field 36
    hook script 61
    scripts 103, 104
ASLEnabled 39

C

calculated values 38
    adding 39
    editing 41
Category 39
    field 36
causality 54, 56
    aggregation 61
    occurring on an element related to the notified element 58
    occurring on an instance related to the notified element 58
    occurring on an instance related to the notified instance 56
    occurring on the same element 57
    occurring on the same instance 56
    tab 55
CheckIsManaged checkbox 34
ClassName 39, 47
    field 35
ClearedByOtherNotifications field 53
ClearedbyOthers tab 53
clearing behavior 51
ClearOnAcknowledge 39
    checkbox 33
ClearsForAllNotifications checkbox 52
ClearsForOtherNotifications field 52
ClearsForOthers tab 52, 53
Close Configuration menu option 76
configuration parameters
    ECI objects 32
    NCI objects 51
copy
    ECI configuration parameters 43
    NCI configuration parameters 62
Create
    EnumValueMapping 79
    EventConfigInfo 78
    NotifConfigInfo 79
    Package 79

D

DedupOn checkbox 33
DedupResendInterval field 33
Description field 36
Discard 62
DiscardTrap 39
DisplayName field 36
documentation
    generating 95

E

ECI Setup dialog box 35, 36
    Active 32
    Calc'd Values tab 38
    ClassName 35
    Event Options tab
        ASL 36
        Description 36
        DisplayName 36
        Event Stream 37
        EventType 36
        Java 37
        Logfile 37
        ServiceName 36
        SysNameOrAddr 36
        UnknownAgent 37
    Event Setup tab
        Category 36
        ElementClassName 35
        ElementName 35
        EventDisplayName 36
        EventName 36
        EventText 36
        Expiration 36
        InstanceName 35
        Severity 36
        State 36
    General tab 32
        CheckIsManaged 34
        ClearOnAcknowledge 33
        DedupOn 33
        DedupResendInterval field 33
        Event Fork 33
        InMaintenance 33
        TimeBasedThresholdActive 33
        TimeBasedThresholdCount 33
        TimeBasedThresholdWindow 33
        UsesEnumWildcard 34
    User Defined tab 38
Edit
    a server's settings menu option 87

calculated values 41
menu options 78
Create EnumValueMapping 79
Create EventConfigInfo 78
Create NotifConfigInfo 79
Create Package 79
ElementClassName 39, 62
field 35
ElementName 39
field 35
EMC Ionix
Adapter Platform (OI) 14
ITOps NOTIF 14
Service Assurance Manager (SAM) 14
SNMP Trap Adapter 15
enumeration, keyword/value pairs 31
EnumValueMapping 70, 71
Event 24
Fork checkbox 33
Options tab 36
Setup tab 35
Stream field 37
Text 60
event 24, 25, 29, 35
matching 28
normalization 26
options values 36
packaging 25
setup values 35
wrapping 26
Event Configuration Information (ECI) objects 16, 17, 18
configuration parameters 32, 71
copying 43
creating 78
enumeration lookups 31
event matching 28
naming 30
naming conventions 28, 29
overview 24
processing 24, 25
substitution values 97, 100
eventAgent 26
eventBaseID 26, 29, 30
EventConfigInfo 70, 71
EventDisplayName 62
EventDisplayName field 36
EventName 39, 47, 62
field 36
eventSub1ID 26
eventSub2ID 26
eventSubID1 30
eventSubID2 30
EventText 39, 62
field 36
eventTimeStamp 26
EventType 39
field 36
Exit menu option 78
Expiration 39
field 36

F

File menu options 73
Close Configuration 76
Exit 78
Generate HTML 78
Import trap_mgr.conf 76
Merge Configuration 75
New Configuration 74
Open Configuration 74
Save Configuration 76
Save Configuration As 76

G

Generate
HTML menu option 78
ITOps NOTIF documentation 95

H

Help menu options 91
About 92
Migration
Import from an older NOTIF server 93
Import from file 92
Substitutions 92

I

Import
from an older NOTIF server menu option 93
from file menu option 92
trap_mgr.conf menu option 20, 76
InMaintenance 39
checkbox 33
InstanceName 39
field 35
ITOps NOTIF
classes 70
EnumValueMapping 70, 71
EventConfigInfo 70
NotifConfigInfo 70, 72
NotifPackage 70, 72
Edit menu options 78
Create EnumValueMapping 79
Create EventConfigInfo 78
Create NotifConfigInfo 79
Create Package 79
Editor
using 70
Event Configuration Information (ECI) objects 16, 17, 18, 24, 25, 28, 29, 30, 32, 71
File menu options 73
Close Configuration 76
Exit 78
Generate HTML 78
Import trap_mgr.conf 76
Merge Configuration 75
New Configuration 74
Open Configuration 74
Save Configuration 76

Save Configuration As 76
generating documentation 95
Help menu options 91
About 92
Migration 92
Import from an older NOTIF server 93
Import from file 92
Substitutions 92
Notification Configuration Information (NCI) objects 16, 17, 24, 45, 46, 47, 50, 51, 72
overview 14
Remote menu options 80
Edit a server's settings 87
Manage connections 88
Push package to server 81, 82
Push to server 81
Reload server 83
Retrieve a configuration file from server 85
Retrieve running configuration from server 84
topology tree view 70
writing
ASL scripts 103, 104
Java classes 103, 107

J

Java
classes 103, 107
field 37
JavaEnabled 39

K

keyword/value pairs 31, 71
Keyword1|Value1 26
Keyword2|Value2 26

L

Logfile field 37

M

Manage connections menu option 88
Merge Configuration menu option 75
migration 92, 93

N

NCI Setup dialog box
Active checkbox 46, 47
AggregatesFrom tab 60
Event Text 60
AggregatesTo tab 60, 61
Causality tab 54, 55
ClearedbyOthers tab 53
ClearedByOtherNotifications 53
ClearsForOthers tab 52, 53
ClearsForOtherNotifications 52
Publish Delay 54
General tab 52
New Configuration menu option 74
normalizing raw events 26
NotifConfigInfo 70, 72
Notification aggregation 60, 61
Notification Configuration Information (NCI) objects 16, 17
ASL hook script 61
causality 54, 61
clearing behavior 51
configuration parameters 51, 72
copying 62
creating 79
naming conventions 50
notification matching 47
overview 46
processing 24, 45, 47
Publish Delay 54
NotifPackage 70, 72

O

occurring
on an element related to the notified element 58
on an instance related to the notified element 58
on an instance related to the notified instance 56
on the same element 57
on the same instance 56
OI 14
OID 31, 71
Open Configuration menu option 74
overview
ECI objects 24
ITOps NOTIF 14
NCI objects 46

P

Publish Delay 54
Push
package to server menu option 81, 82
to server menu option 81

R

raw event 24, 25, 29, 35
normalization 26
Reload server menu option 83
Remote menu options 80
Edit a server's settings 87
Manage connections 88
Push package to server 81, 82
Push to server 81
Reload server 83
Retrieve a configuration file from server 85
Retrieve running configuration from server 84
Retrieve
a configuration file from server menu option 85
running configuration from server menu option 84

S

Save
Configuration As menu option 76

Configuration menu option 76
Service Assurance Manager (SAM) 14
ServiceName field 36
Severity 39, 62
field 36
SNMP Trap Adapter 15
State 39
field 36
Substitution 97
substitution values 32, 39, 97, 100
Substitutions menu option 92
SysNameOrAddr field 36

T

TimeBasedThresholdActive checkbox 33
TimeBasedThresholdCount field 33
TimeBasedThresholdWindow field 33
trap_mgr.conf 14, 16, 20, 25, 76
trapd.conf 16

U

UnknownAgent field 37
UpdateUserDefinedFields 39
Use Enum Wildcard 31
user defined values 38
UserDefined1 - 20 39, 62
UsesEnumWildcard checkbox 34

V

values
calculated 38
substitution 32, 39
user defined 38

W

wrapping raw events 26