Alcatel-Lucent 5620

SERVICE AWARE MANAGER | RELEASE 9.0 R6

SCRIPTS AND TEMPLATES DEVELOPER GUIDE

3HE 06499 AAAF TQZZA Edition 01

Disclaimers

Alcatel-Lucent products are intended for commercial uses. Without the appropriate network design engineering, they must not be sold, licensed or otherwise distributed for use in any hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life-support machines, or weapons systems, in which the failure of products could lead directly to death, personal injury, or severe physical or environmental damage. The customer hereby agrees that the use, sale, license or other distribution of the products for any such application without the prior written consent of Alcatel-Lucent, shall be at the customer's sole risk. The customer hereby agrees to defend and hold Alcatel-Lucent harmless from any claims for loss, cost, damage, expense or liability that may arise out of or in connection with the use, sale, license or other distribution of the products in such applications.

This document may contain information regarding the use and installation of non-Alcatel-Lucent products. Please note that this information is provided as a courtesy to assist you. While Alcatel-Lucent tries to ensure that this information accurately reflects information provided by the supplier, please refer to the materials provided with any non-Alcatel-Lucent product and contact the supplier for confirmation. Alcatel-Lucent assumes no responsibility or liability for incorrect or incomplete information provided about non-Alcatel-Lucent products.

However, this does not constitute a representation or warranty. The warranties provided for Alcatel-Lucent products, if any, are set forth in contractual documentation entered into by Alcatel-Lucent and its customers.

This document was originally written in English. If there is any conflict or inconsistency between the English version and any other version of a document, the English version shall prevail.

Preface

The Preface provides general information about the 5620 Service Aware Manager documentation suite, including this guide.

Prerequisites

Readers of the 5620 SAM documentation suite are assumed to be familiar with the following:

- 5620 SAM software structure and components
- 5620 SAM GUI operations and tools
- typical 5620 SAM management tasks and procedures
- device and network management concepts

5620 SAM documentation suite

The 5620 SAM documentation suite describes the 5620 SAM and the associated network management of its supported devices. Contact your Alcatel-Lucent support representative for information about specific network or facility considerations.

Table 1 lists the documents in the 5620 SAM customer documentation suite.

Table 1 5620 SAM customer documentation suite

Guide	Description
5620 SAM core documentation	
5620 SAM Release Description	The *5620 SAM Release Description* provides information about the new features associated with a 5620 SAM software release.

(1 of 4)

Guide	Description
5620 SAM Planning Guide	The *5620 SAM Planning Guide* provides information about 5620 SAM scalability and recommended hardware configurations.
5620 SAM System Architecture Guide	The *5620 SAM System Architecture Guide* is intended for technology officers and network planners to increase their knowledge of the 5620 SAM software structure and components. It describes the system structure, software components, and interfaces of the 5620 SAM. In addition, 5620 SAM fault tolerance, security, and network management capabilities are discussed from an architectural perspective.
5620 SAM | 5650 CPAM Installation and Upgrade Guide	The *5620 SAM | 5650 CPAM Installation and Upgrade Guide* provides OS considerations, configuration information, and procedures for the following: • installing, upgrading, and uninstalling 5620 SAM and 5650 CPAM software in standalone and redundant deployments • 5620 SAM system migration to a different system • conversion from a standalone to a redundant 5620 SAM system
5620 SAM User Guide	The *5620 SAM User Guide* provides information about using the 5620 SAM to manage the service-aware IP/MPLS network, including GUI basics, commissioning, service configuration, and policy management. The *5620 SAM User Guide* uses a task-based format. Each chapter contains: • a workflow that describes the steps for configuring and using the functions • detailed procedures that list the configurable parameters on the associated forms 5620 SAM management information specific to LTE network elements is covered in the *5620 SAM LTE ePC User Guide* and *5620 SAM LTE RAN User Guide*. 5620 SAM management information specific to 1830 PSS network elements is covered in the *5620 SAM Optical User Guide*.
5620 SAM Integration Guide	The *5620 SAM Integration Guide* provides procedures to allow the 5620 SAM to integrate with additional components.
5620 SAM Supervision Module User Guide	The *5620 SAM Supervision Module User Guide* provides information about how to configure and use the web-based 5620 SAM Supervision Module for fault management and at-a-glance network element monitoring.
5620 SAM Scripts and Templates Developer Guide	The *5620 SAM Scripts and Templates Developer Guide* provides information that allows you to develop, manage, and execute CLI-based or XML-based scripts or templates. The guide is intended for developers, skilled administrators, and operators who are expected to be familiar with the following: • CLI scripting, XML, and the Velocity engine • basic scripting or programming • 5620 SAM functions
5620 SAM Parameter Guide	The *5620 SAM Parameter Guide* provides: • parameter descriptions that include value ranges and default values • parameter options and option descriptions • parameter and option dependencies • parameter mappings to the 5620 SAM-O XML equivalent property names There are dynamic links between the procedures in the *5620 SAM User Guide* and the parameter descriptions in the *5620 SAM Parameter Guide*. Parameters specific to LTE network elements are covered in the *5620 SAM LTE Parameter Reference*. Parameters specific to 1830 PSS network elements are covered in the *5620 SAM Optical Parameter Reference*.
5620 SAM Statistics Management Guide	The *5620 SAM Statistics Management Guide* provides information about how to configure performance and accounting statistics collection and how to view counters using the 5620 SAM. Network examples are included.

(2 of 4)

Guide	Description
5620 SAM Maintenance Guide	The *5620 SAM Maintenance Guide* provides procedures for: • generating baseline information for 5620 SAM applications • performing daily, weekly, monthly, and as-required maintenance activities for 5620 SAM-managed networks

5620 SAM Troubleshooting Guide	The *5620 SAM Troubleshooting Guide* provides task-based procedures and user documentation to:
	• help resolve issues in the managed and management networks
	• identify the root cause and plan corrective action for:
	  • alarm conditions on a network object or customer service
	  • problems on customer services with no associated alarms
	• list problem scenarios, possible solutions, and tools to help check:
	  • network management LANs
	  • network management platforms and operating systems
	  • 5620 SAM client GUIs and client OSS applications
	  • 5620 SAM servers
	  • 5620 SAM databases

5620 SAM Alarm Reference	The *5620 SAM Alarm Reference* provides a list of all alarms that the 5620 SAM can raise. The reference is organized by network element type.
5620 SAM Glossary	The *5620 SAM Glossary* defines terms and acronyms used in all of the 5620 SAM documentation, including 5620 SAM LTE documentation.
5620 SAM Network Element Compatibility Guide	The *5620 SAM Network Element Compatibility Guide* provides release-specific information about the compatibility of managed device features in 5620 SAM releases.
5620 SAM LTE documentation	
5620 SAM LTE RAN Release Description	The *5620 SAM LTE RAN Release Description* provides information about the LTE RAN features associated with the release.
5620 SAM LTE ePC User Guide	The *5620 SAM LTE ePC User Guide* describes how to discover, configure, and manage LTE ePC devices using the 5620 SAM. The guide is intended for LTE ePC network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire *5620 SAM LTE ePC User Guide* before you attempt to use the 5620 SAM in your LTE network.
5620 SAM LTE RAN User Guide	The *5620 SAM LTE RAN User Guide* describes how to discover, configure, and manage the Evolved NodeB, or eNodeB, using the 5620 SAM. The guide is intended for LTE RAN network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire *5620 SAM LTE RAN User Guide* before you attempt to use the 5620 SAM in your LTE network.
5620 SAM LTE Parameter Reference	The *5620 SAM LTE Parameter Reference* provides a list of all LTE ePC and LTE RAN parameters supported in the 5620 SAM.
5620 SAM-O documentation	
5620 SAM XML OSS Interface Developer Guide	The *5620 SAM XML OSS Interface Developer Guide* provides information that allows you to: • use the 5620 SAM XML OSS interface to access network management information • learn about the information model associated with the managed network • develop OSS applications using the packaged methods, classes, data types, and objects necessary to manage 5620 SAM functions
5620 SAM 3GPP OSS Interface Developer Guide	The *5620 SAM 3GPP OSS Interface Developer Guide* describes the components and architecture of the 3GPP OSS interface to the 5620 SAM. It includes procedures and samples to assist OSS application developers to use the 3GPP interface to manage LTE devices.

(3 of 4)

Guide	Description
5620 SAM 3GPP OSS Interface Compliance Statements	The *5620 SAM 3GPP OSS Interface Compliance Statements* document describes the compliance of the 5620 SAM 3GPP OSS interface with the 3GPP standard.
5620 SAM optical documentation	
5620 SAM Optical User Guide	The *5620 SAM Optical User Guide* describes how to discover, configure, and manage optical devices using the 5620 SAM. The guide is intended for optical network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire *5620 SAM Optical User Guide* before you attempt to use the 5620 SAM in your network.
5620 SAM Optical Parameter Reference	The *5620 SAM Optical Parameter Reference* provides a list of all optical device parameters supported in the 5620 SAM.

(4 of 4)

Obtaining customer documentation

You can obtain 5620 SAM customer documentation:

- from the product
- on the web

On-product documentation

The 5620 SAM on-product customer documentation is delivered in HTML and PDF. Choose Help→User Documentation from the 5620 SAM client GUI to open the help system in a web browser.

The help system opens to the User Documentation Index, which provides a summary of and links to all 5620 SAM customer documents.

Click on the Using the help system tab on the User Documentation Index page to find usage tips for navigating and searching within the on-product customer documentation.

You can return to the User Documentation Index at any time by clicking on the Home icon, shown in Figure 1.

Figure 1 Home icon

Documentation on the web

The 5620 SAM customer documentation is available for download in PDF format from the Alcatel-Lucent Customer Support Center: http://www.alcatel-lucent.com/myaccess. If you are a new user and require access to this service, please contact your Alcatel-Lucent support representative.

In addition to the guides listed in Table 1, Release Notices and other documents not delivered on-product are posted to this site.

Working with PDFs

You can download PDFs of individual guides from the Alcatel-Lucent Customer Support Center, or you can choose to download a zip of all PDFs for a particular release.

You can use the Search function of Acrobat Reader (File→Search) to find a term in a PDF of any 5620 SAM document. To refine your search, use appropriate search options (for example, search for whole words only or enable case-sensitive searching). You can also search for a term in multiple PDFs at once, provided that they are located in the same directory. For more information, see the Help for Acrobat Reader.

Cross-book hotlinks, for example, from a parameter name in the *5620 SAM User Guide* to a description of that parameter in the *5620 SAM Parameter Guide*, work only if both PDF files are in the same directory.

Note — Users of Mozilla browsers may receive an error message when opening the PDF files in the 5620 SAM documentation suite. The offline storage and default cache values used by the browsers are the cause of the error message.

Alcatel-Lucent recommends changing the Mozilla Firefox offline storage or Mozilla 1.7 cache value to 100 Mbytes to eliminate the error message.

Documentation conventions

Table 2 lists the conventions that are used throughout the documentation.

Table 2 Documentation conventions

Convention	Description	Example
Key name	Press a keyboard key	Delete
Italics	Identifies a variable	*hostname*
Key+Key	Type the appropriate consecutive keystroke sequence	CTRL+G
Key-Key	Type the appropriate simultaneous keystroke sequence	CTRL-G
*	An asterisk is a wildcard character, which means “any character” in a search argument.	log_file*.txt
↵	Press the Return key	↵
—	An em dash indicates there is no information.	—
→	Indicates that a cascading submenu results from selecting a menu item	Policies→Alarm Policies

Procedures with options or substeps

When there are options in a procedure, they are identified by letters. When there are substeps in a procedure, they are identified by Roman numerals.

Example of options in a procedure

At step 1, you can choose option a or b. At step 2, you must do what the step indicates.

1 This step offers two options. You must choose one of the following.

   a This is one option.

   b This is another option.

2 You must perform this step.

Example of substeps in a procedure

At step 1, you must perform a series of substeps within a step. At step 2, you must do what the step indicates.

1 This step has a series of substeps that you must perform to complete the step. You must perform the following substeps.

   i This is the first substep.

   ii This is the second substep.

   iii This is the third substep.

2 You must perform this step.

Measurement conventions

Measurements in this document are expressed in metric units and follow the Système international d'unités (SI) standard for abbreviation of metric units. If imperial measurements are included, they appear in brackets following the metric unit.

Table 3 lists the measurement symbols used in this document.

Table 3 Bits and bytes conventions

Measurement	Symbol
bit	b
byte	byte
kilobits per second	kb/s

Important information

The following conventions are used to indicate important information:

Warning — Warning indicates that the described activity or situation may, or will, cause equipment damage or serious performance problems.

Caution — Caution indicates that the described activity or situation may, or will, cause service interruption.

Note — Notes provide information that is, or may be, of special interest.

Contents

Common functionality

Script and template management

Getting started

1 — Overview

1 — Overview

1.1 Introduction

The script management and template management components of the 5620 SAM allow users to develop, manage, and execute customized provisioning actions to simplify the operation of their 5620 SAM network. The 5620 SAM scripts and templates use a common infrastructure with the template feature set providing additional capabilities for service and tunnel provisioning primarily, and support for some equipment. Use scripts to apply to common capabilities and templates where features are unique to the tunnel, service, or equipment template features.

1.2 Script management overview

The 5620 SAM GUI provides the framework that allows users to develop scripts to create, delete, configure, or view objects in the 5620 SAM network. Script management is broken into two functional areas:

- CLI scripting – allows operators to execute CLI scripts without knowing the underlying CLI commands
- XML API scripting – allows operators to execute more complex commands using XML

Both CLI and XML API support the Velocity engine for complex scripting.

Script management can be used to:

- create, configure, and test scripts
- view and compare versions of scripts
- preview scripts
- start and stop the execution of scripts associated with managed NEs
- view, compare, and store the results of executions of individual and multiple scripts
- create a customized user-defined GUI

See chapter 4 for more information.

1.3 Template management overview

A template is an object managed by the 5620 SAM that is used as a starting point for configuring complex managed objects such as services and tunnels. Templates allow users to define common characteristics that can be shared by multiple managed objects by associating them with a template instance at creation time.

Template management is divided into two functional areas:

- service templates
- tunnel templates

Service templates

You can use service templates to facilitate service creation, including the service itself and sub-components such as service sites, access interfaces, and SDP bindings. See chapter 6 for more information.

Tunnel templates

You can use tunnel templates to facilitate LSP and SDP tunnel creation. Tunnel types include dynamic LSPs, LSP paths, P2MP dynamic LSPs, and service tunnels.

See chapter 6 for more information.

Common functionality

2 — Scope of command

2.1 Introduction

This chapter provides information about functionality that is common to script management and template management.

2.2 Script management

Access to the script manager is controlled by the admin, script management, and script execution scope of command roles. Users that have the admin or script management role can create, modify, delete, and execute scripts. Users that have the script execution role can view and run scripts, view and save results, configure script targets and instances, and view historical results.

In addition to the scope of command access roles and profiles, access to the script manager is further defined by span of control. Span of control profiles determine where a user can apply scope of command access roles and profiles. For example, a user may have privileges to execute scripts, however a span of control profile defines which scripts the user can execute.

Script execution client permissions

Script execution from a single-user client station requires a user account that has read, write and execute permissions for the following directories:

- *install_dir*/nms/bin/clientCache on Solaris
- *install_dir*\nms\bin\clientCache on Windows

where *install_dir* is the 5620 SAM client installation location, typically /opt/5620sam/client on Solaris and C:\5620sam\client on Windows

Script execution from a client delegate station requires a user account that has read, write, and execute permissions for the following directory on the client delegate server station:

- *home_dir*/5620sam/clientCache
  where *home_dir* is the UNIX home directory of the user

Only users that are assigned the mirror service management scope of command role can create, modify, delete, or view mirror services and mirror service templates.

2.3 Templates

Tunnel and service template configuration follow the same rules. The user must have the appropriate roles or read-write permissions on tunnel and service components. Only users that are assigned the template script management scope of command role can manage XML API tunnel and service template scripts.

3 — Creating and executing scripts

3.1 Introduction

This chapter provides information about the 5620 SAM functionality that supports the development of CLI and XML API scripts and templates.

3.2 Script Editor

Scripts and templates use the same Script Editor to develop scripts or templates. You can use the Script Editor to modify the script or template GUI and code. See Procedures 3-1 and 3-2 for information about how to open the Script Editor.

By default, the Script Editor displays a split view of the GUI Builder and code windows. You can change the layout of windows by clicking on the Show Code Only button or the Show GUI Builder Only button. You can also choose to have the Property Selector form open. It is an optional window that is only applicable for XML API configuration templates.

Procedure 3-1 To open the Script Editor for scripts

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Choose a script type from the object drop-down menu.

3 Perform one of the following:

a Create a new script:

i Click on the Create button to create a script. The *Script_type* Script (Create) form opens with the General tab displayed.

ii Configure the parameters, as required.

Note 1 — When creating a CLI script, you must set the Content Type parameter to Velocity.

Note 2 — An asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

b Open an existing script:

i Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

ii Choose a script and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

iii Configure the parameters, as required.

Note — An asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

4 Click on the Versions tab button.

5 Click on the Create button to create a script or double-click on a script to modify an existing script. The Script Editor form opens.

Procedure 3-2 To open the Script Editor for templates

1 Perform one of the following.

- **a** Use the Manage Templates form. Choose Manage→Templates from the 5620 SAM main menu. The Manage Templates list form opens.
- **b** Use the Script Manager. Perform the following steps.
  - **i** Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.
  - **ii** Choose the Template (Scripting) object from the drop-down menu.

2 Perform one of the following.

- **a** Create a new template. Perform the following steps.
  - **i** Click on the Create button to create a template. The Template (Create) form opens with the General tab displayed.
  - **ii** Configure the parameters, as required.

Note — An asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

- **b** Open an existing template. Perform the following steps.
  - **i** Configure the list filter criteria. A list of templates appears at the bottom of the Manage Templates form.
  - **ii** Choose a template and click on the Properties button. The Template (Edit) form opens with the General tab displayed.
  - **iii** Configure the parameters, as required.

Note — An asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

3 Click on the Apply button. A dialog box appears.

4 Click on the OK button. A Template (Edit) form opens. The Script Editor option appears under the More Actions button.

5 Click on the More Actions button and choose Script Editor. The Script Editor form opens.

3.3 Script format

The script format for CLI and XML scripts consists of the following:

- header—typically includes descriptions and comments
- velocity GUI properties—these properties include the GUI components that are generated by the script
- commands—includes CLI commands for CLI scripts, or XML API commands for XML API scripts

Table 3-1 describes the CLI script format and Table 3-2 describes the XML API script format.

Table 3-1 CLI script format

Format	Example
Header	#################### ## Description: ## ####################
Velocity GUI properties	#* <velocityProperties> </velocityProperties> *#
CLI commands	show service VPLS 3

Table 3-2 XML API script format

Format	Example
Header	#################### ## Description: ## ####################
Velocity GUI properties	#* <velocityProperties> </velocityProperties> *#

(1 of 2)

Format	Example
XML API commands	<xmlapiRequest xmins=”xmlapi_1.0”> </xmlapiRequest>

(2 of 2)

3.4 Velocity overview

Velocity is a template tool that provides a template language to reference objects that are defined in another language. The Velocity engine allows you to add complex algorithms to CLI and XML API scripts. Table 3-3 describes the syntax for Velocity-based CLI and XML API scripts.

Table 3-3 Velocity script syntax

Tag	Description
##comments *#*comments*#*	Specifies that the text that follows is treated as comments. Comments are ignored by the 5620 SAM when it generates commands from the script and are removed from the script before the script is sent to the NE or XML API.
$*parameter* $*parameter.attribute* $*parameter.method()*	The $ tags indicate Velocity parameters, which can include primitive values or complex objects.

Warning — Velocity automatically removes excess whitespace from scripts. For example, when the following script is run:

```
show port #if($siteA)$siteA#end

show port #if($portA)$portA#end
```

the script is converted to:

```
show port 1/1/1show port 2/2/2
```

The tag "#end" removes all whitespace after the tag, including new lines and carriage returns. This causes the CLI script to execute as one command to the NE. Use one of the following workarounds to prevent whitespace from being removed. Both workarounds move the #end command to the next line.

```
show port #if($siteA)$siteA

#end

show port #if($portA)$portA

#end
```

or

```
#if($siteA)

 show port $siteA

#end

#if($portA)

 show port $portA

#end
```

See http://velocity.apache.org for more information.

Velocity parameters that you specify in a Velocity script are replaced by the system, script, or user, when the script is executed.

Note 1 — If you import a script into the Script Editor, you may be required to manually replace certain text with tags.

Note 2 — The 5620 SAM script manager does not perform a syntax analysis on scripts.

Note 3 — If you define a long or short value that is to be used in a Velocity "foreach" statement, you must first convert the value to an integer value with the "intValue()" method.

For example, the following script is unsuccessful:

```
#foreach ($svcId in [$svcIdFrom..$svcIdTo])

## ...

#end
```

The following script first converts the value to an integer value:

```
#foreach ($svcId in

 [$svcIdFrom.intValue()..$svcIdTo.intValue()])

## ...

#end
```

Where svcIdFrom and svcIdTo are defined as type "long".

You can include or parse other scripts in the scripts you create. The following is the Velocity format:

```
#include ("script_name")

#parse ("script_name")
```

The script management format also contains the version. The following is an include/parse command for an XML API script:

```
#include ("<script_name>:xmlversion-<script_version>")

#parse ("<script_name>:xmlversion-<script_version>")
```

The following is an include/parse command for a CLI script:

```
#include ("<script_name>:version-<script_version>")

#parse ("<script_name>:version-<script_version>")
```

For more information about the Velocity engine and Velocity template language and parameters, see http://velocity.apache.org/engine/devel/user-guide.html and http://velocity.apache.org/engine/devel/vtl-reference-guide.html.

See Table 4-1 for a description of the syntax for CLI scripts that are not Velocity-based.

3.5 User interface generation

CLI and XML API scripts contain a UI generation section where you can include a Velocity header that adds a user interface to your scripts. The user interface can include tabs, drop-down menus, and validation text fields.

You can also use the GUI Builder to create a UI and generate a Velocity header. See section "Velocity GUI Builder" for more information.

Code 3-1 shows a sample Velocity header for a generated UI.

Code 3-1: Sample Velocity UI header

```
#*<velocityProperties>
   <tab>
      <uiName>General</uiName>
      <uiOrder>-1</uiOrder>
      <tooltip>The general tab</tooltip>
      <name>General</name>
      <group>
         <uiOrder>-1</uiOrder>
         <tooltip>The general group</tooltip>
         <xcoord>0</xcoord>
         <name>General</name>
         <ycoord>0</ycoord>
         <property>
            <uiName>Profile Name:</uiName>
            <type>string</type>
            <uiOrder>4</uiOrder>
            <required>true</required>
            <tooltip>The unique name of the profile</tooltip>
            <mode>modify</mode>
            <default>private</default>
            <max>0</max>
            <ycoord>0</ycoord>
            <min>0</min>
            <xcoord>0</xcoord>
            <name>name</name>
         </property>
         <property>
            <uiName>Password:</uiName>
            <type>string</type>
            <uiOrder>2</uiOrder>
            <required>true</required>
            <tooltip>The password</tooltip>
            <mode>modify</mode>
            <default>admin</default>
            <max>10</max>
            <ycoord>0</ycoord>
            <min>3</min>
            <xcoord>0</xcoord>
            <name>password</name>
         </property>
         <property>
            <uiName>User Name:</uiName>
            <type>string</type>
```

```
            <uiOrder>0</uiOrder>
            <required>true</required>
            <tooltip>The user name</tooltip>
            <mode>modify</mode>
            <default>admin</default>
            <max>0</max>
            <ycoord>0</ycoord>
            <min>0</min>
            <xcoord>0</xcoord>
            <name>username</name>
         </property>
         <property>
            <uiName>Unique ID:</uiName>
            <type>range</type>
            <uiOrder>1</uiOrder>
            <tooltip>The unique id</tooltip>
            <mode>modify</mode>
            <default>1001</default>
            <max>10000</max>
            <ycoord>0</ycoord>
            <min>3</min>
            <xcoord>0</xcoord>
            <name>id</name>
         </property>
      </group>
   </tab>
</velocityProperties>
*#
```

Figure 3-1 shows the instance configuration form with the UI that is generated from the sample Code 3-1 in an XML API script.

Figure 3-1 Sample generated UI

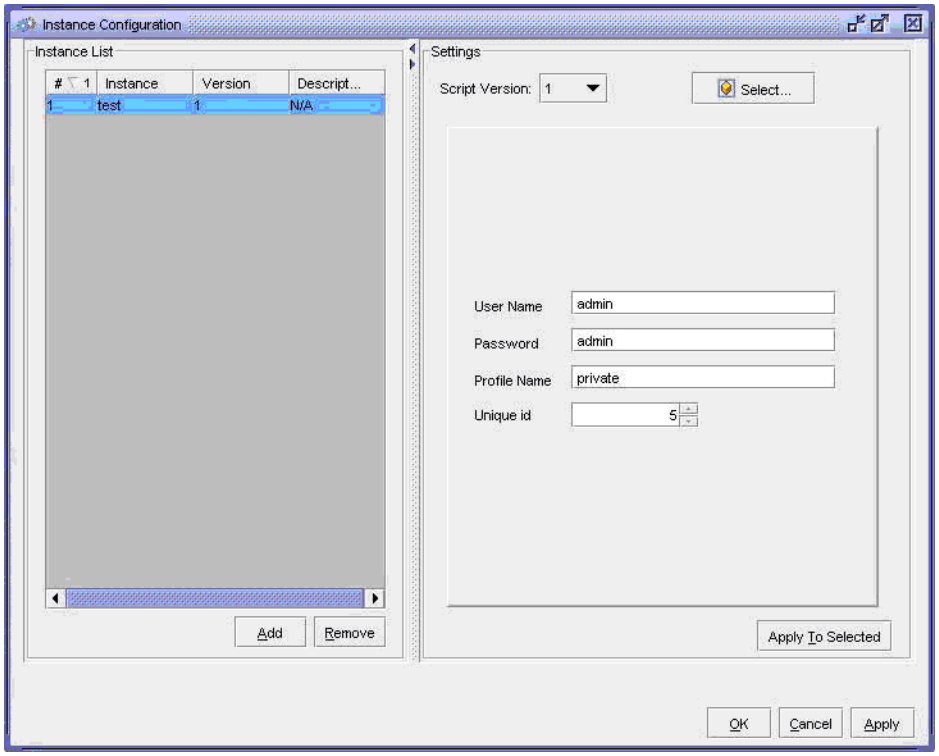


Table 3-4 describes the Velocity tags that are supported for generating a UI.

Table 3-4 Velocity tags for generating a UI

Tag	Value	Description

velocityProperties	—	Identifies the start of the UI definition
		The tag must be contained in Velocity comments. For example:
		`#* <velocityProperties>`
		`...`
		`</velocityProperties>*#`

property	—	Identifies a property that should be displayed to the user

(1 of 3)

Tag	Value	Description
type	string largeString integer long float format combobox (drop-down menu) range date bit mask password boolean selection multiselection	Identifies the type of the property
default	—	Identifies the default value for the property If the type is a list, the default value is the name of the default item in the list.
uiOrder	—	Identifies the order in which fields are displayed in the UI
list	—	The list of items to display in the list (combo box type) Include the item tags as a list.
item	—	An item in the drop-down menu The item can contain a name and value. The name is displayed in the list. The value tag is returned for substitution.
name	—	The name of the property The property name must be identical to the name of the Velocity parameter (without $).
uiName	—	The name to display in the UI
value	—	The value of the Velocity parameter (only used for items in a combo box)
tooltip	—	The tool tip to display to the user about the property

(2 of 3)

Tag	Value	Description
format	—	Used for a special format field and uses a MaskFormatter. The following are the text mask characters for the MaskFormatter class: • # Any number • U Any character. Lowercase characters are mapped to uppercase characters • L Any character. Uppercase characters are mapped to lowercase characters • A Any character or number • _ Use the underscore to represent any character • % Any number, character, or symbol • H Any hexidecimal character: • 0 to 9 • a to f • A to F
range	—	A range of numeric values The range contains a min and max tag that are subelements that define the range.
min	—	Minimum value for the range If the type tag is a string, the min tag is the minimum length. If the type tag is a number, the min tag is the lowest value.
max	—	Maximum value for the range If the type tag is a string, the max tag is the maximum length. If the type tag is a number, the max tag is the highest value.
runtime	—	Identifies that the property is not stored. The property is specified at execution time.
required	—	Identifies whether the property must have a value entered
group	—	Used for grouping properties in the UI The group tag supports the name and tool tip properties.
tab	—	Used for adding tabs on the UI to show groups The tab tag supports the name and tool tip properties.

(3 of 3)

3.6 Velocity GUI Builder

The Script Editor for CLI Velocity and XML API scripts includes a GUI Builder which allows you to create the XML content for the Velocity properties. You can use the GUI Builder to create or modify Velocity properties, or test the generated UI. You can specify all Velocity property types, and configure the properties using the GUI Builder form, which includes the following:

- properties tree
- property attribute list
- preview window

The properties tree lists the hierarchy of components that are added, which includes tabs, groups, and properties. Each property in the tree is identified by a Velocity name tag. When properties are inserted, they are appended to the parent that is selected in the tree. When you do not select a parent, the property is added to the first parent in the list.

The property attribute list contains the attributes that can be configured for a selected property in the properties tree. The attributes can include group or tab placement, tool tips, and X and Y coordinates. Depending on the parameter, the value format for editing may differ. For example, a Boolean parameter uses a check box. Other values, such as filters, list actions.

The preview window displays the generated UI based on the configuration of the properties. The generated UI preview changes dynamically when you change property values.

Velocity properties

Velocity property components are used in scripts to allow the user to provide input. The values provided by the user are then made available to the script as Java objects. For example, a String component returns a java.lang.String. These values are then accessed within the script using the component name, for example $myString. When the script is previewed or executed, the name is replaced with a string representation of the object, for example the result of toString().

Any other methods for that object type are also available. All components except bitmask return standard Java objects. A developer can use the Java documentation to find the available methods.

Container components are used to contain and organize components. The components can be configured at design time, and altered via actions at run time, but the components do not return any values to use in the script.

Table 3-5 lists and describes the properties that you can add to a GUI.

Table 3-5 Velocity properties

Icon	Property	Description	XML tag	Dependencies
Container components				

(1 of 3)

Icon	Property	Description	XML tag	Dependencies
	Tab	Inserts a tab button. This is the top level container, and contains group components. All scripts must have at least one tab. This property allows common attributes to be displayed in their own page on the GUI.	<tag></tag>	—
	Group	Inserts a panel on a tab in which you can group other properties. Provides containers for the other components, except tabs. The tabs need at least one group, and the other components must be within a group. If a group has a displayed name, the border appears. If a displayed name is not displayed, the components appear to be contained by the tab.	<group></group>	Tab
String components				
	String	Inserts a text box in a group. Suitable for single-line strings (up to 252 characters), such as names and short descriptions.	<type>string</type>	Group
	Large text	Inserts a large text box. Suitable for multi-line strings and contains up to 4000 characters.	<type>largestring</type>	
	Password	Inserts a password. This works the same as the string component, except that the entered value is hidden.	<type>password</type>	
	Format	Inserts a format field. This is a string component with an option to force the user to enter text in a specific format, such as a phone number or IP address. Formats can be defined using either a mask or a regular expression. To use a regular expression, the useRegExp (“RegularExpression?”) property must be set to true.	<type>format</type>	
Numeric property components				
	Integer	Inserts an integer field. Accepts and returns integers between -231(-2147483648) and 231-1(2147483647).	<type>integer</type>	Group
	Long	Inserts a long field. Accepts and returns longs between -263(-9223372036854775808) and 263-1(9223372036854775807).	<type>long</type>	
	Float	Inserts a float field. Accepts and returns floats. For limits, see the Java documentation for java.lang.Float.	<type>float</type>	
Object selection property components				

(2 of 3)

Icon	Property	Description	XML tag	Dependencies
	Selection	Inserts a panel with a Select button to choose an object from a list form. Used by the operator to choose a 5620 SAM object from a list. The property is configured at design time by providing a search class and filter, and returns a 5620 SAM object. All properties for objects have an accessor method that returns the value of the property. The name of the accessor method is getPropertyName, where PropertyName is the name of the object property. For example, getObjectFullName returns the value of the objectFullName property of the object. For the properties that are available on each object, see the XML Reference.	<type>selection</type>	Group
	Multi-selection	Inserts a panel with an Add button to choose several objects from list forms. This is similar to the selection component, except that it allows the operator to choose more than one object. It returns an ArrayList? of 5620 SAM objects, and so any meaningful access to the selected objects is via list methods.	<type>multiselection</type>	
	Combobox	Inserts a combo box. A combo box is used to choose from a list of values. It returns the value of the item selected by the operator as a String.	<type>combobox</type>	
	Checkbox	Inserts a check box. A check box is used to select between boolean true and false values.	<type>boolean</type>	
Other property components				
	Date/time	Inserts a date and time type field and a pop-up calendar. Used to specify a date. The property displays the date and time as a text string that can be modified using spinner buttons, or a pop-up calendar for date selection. It returns a java.util.Date object, which can be accessed using any of the Date methods, or by using the Velocity DateTool?.accessible via $vmDate.	<type>date</type>	Group
	Bit mask	Inserts a bit mask component. Used to choose 0 or more items from a list. The property consists of several check boxes, each which represents one of the available items. Unlike the other components that use classes from the standard Java library, the bitmask component returns a VelocityBitmaskHelper?, which is proprietary.	<mask> <bit> <uiName/> <tooltip/> <value/> </bit> <bit> <uiName/> <tooltip/> <value/> </bit> </mask>	

(3 of 3)

Table 3-6 describes the attributes that you can configure for Velocity properties.

Table 3-6 Velocity property attributes

Attribute	Description	Applicable properties	XML tag	Dependencies
Type	Identifies the type of the property	string largeString integer long float format combobox (drop-down menu) range date bit mask password boolean selection multiselection	<type></type>	—
Name	The Velocity name of the property	All properties	<name></name>	—
Display Name	The name displayed in the UI	All properties	<uiName></uiName>	
Tooltip	The tool tip displayed to the user	All properties except Tab	<tooltip></tooltip>	
X Coordinate	The enhanced order on the X-coordinate for components in tabular format	All properties except Tab	<xcoord></xcoord>	Cannot be configured unless the UI Order attribute is set to -1.
Y Coordinate	The enhanced order on the Y-coordinate for components in tabular format	All properties except Tab	<ycoord></ycoord>	Cannot be configured unless the UI Order attribute is set to -1.
Collapsible Group	When enabled, template panels can be collapsed to hide areas that are not of interest to the specific user or expanded to display all areas in a template.	All groups	<useCollapsible>true</useCollapsible>	Set the Display Name attribute to a non-empty string.
UI Order	The order in which components are displayed on the UI	All components	<uiorder></uiorder>	Must be set to -1 when you configure the X Coordinate and Y Coordinate attributes

(1 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Mode	Identifies when a component value must be modified, even if a default value is defined. The options are: • Read-Only • Must Modify • Modify (default) • Hidden	String Float Long Integer Date bit mask Large text Combobox Checkbox Password Format Selection Multiselection	<mode></mode>	Cannot specify Hidden if the Required? attribute is configured
Required?	Identifies whether the component value must be configured by the user	String Long Integer Date Bit mask Large text Combobox Password Format Selection Multiselection	<required></required>	Cannot enable when the Mode attribute is set to Hidden
Runtime?	Identifies whether the component value is specified and applied at execution, and is not stored	String Float Long Integer Date Bit mask Large text Combobox Checkbox Password Format Selection Multiselection	<runtime></runtime>	—

(2 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Action List	Identifies the action that is called and invoked when a component is changed; for example, a button is clicked on, or an option is chosen from a combo box. Table 3-7 describes the actions.	String Float Long Integer Date Bit mask Large text Combobox Checkbox Password Format	<action></action>	—
Unset	Displays an unset check box beside the component. The value of the unset check box can be accessed from the script by adding the special tag $tp_propertyName.getUnset(), where propertyName must be replaced with the actual Name of the Velocity property. Example of script using the unset feature: <description unset="$tp_description.getUnset()">$description</description>	All properties	<unset></unset>	—
Impact	Shows an icon next to the component label to indicate the possible impact of the specified attribute to the network element state. Use if the change of a property causes reset of the network element. The values are: noReset, partialReset, fullReset.	All properties	<impact></impact>	—
Default	The default value of the property	String Float Long Integer Date bit mask Large text Combobox Checkbox Format Selection	<default></default>	—

(3 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Check Box Value – UI Name Check Box Value	Check Box Value - UI Name and Check Box Value work together to provide an option for the user to set the component value to a default, instead of entering it into the component. Both attributes must be set in order to work. Check Box Value - UI Name is the label assigned to the check box in the GUI. When the check box is checked, the rest of the component is dimmed and the value for the component is set to the value of the Check Box Value attribute.	String Float Long Integer Large text Combobox Password Format	<valueCheckBoxUIName> </valueCheckBoxUIName> <valueCheckbox> </valueCheckbox>	—
Minimum	Identifies the minimum value for the component. A minimum value for string and large text, and password properties is the minimum length.	String Float Long Integer Date Large text Combobox Password	<min></min>	—
Maximum	Identifies the maximum value for the component. A maximum value for a string, large text, and password properties is the maximum length. For a format field, the maximum value is used only for display purposes.	String Float Long Integer Date Large text Combobox Password Format	<max></max>	—
Mask	Bitmask value definition that identifies all the bits that can be set in the mask. Each bit can have a uiName, tool tip, and the value. The value is the only required tag. If the uiName is not defined, the value becomes the label for the component.	Date Bit mask	<bit> <name></name> <uiName></uiName> <value>true/false</value> </bit>	—

(4 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Format	Identifies the format that the user can apply to enter in a component. There are two format definitions, mask mode or regular expression. In mask mode, the script writer defines a mask using a combination of constant characters which appear as part of the displayed string, and special characters that act as place holders for user-entered text. For example, a mask for a phone number would look like this: (###)###-### The constant characters are included in the return value, along with the user-entered text. The format component will be in mask mode if the regular expression attribute is not selected. See the Regular Expression attribute in this table for more information.	Format	<format></format>	—
Use Current Time	Identifies whether the current time is used at time of script instantiation	Date	<currentTime>true/false</currentTime>	—
Regular Expression?	Allows a regular expression to be used in a format component. In this mode, the script writer defines the required input in the format field using a regular expression. Unlike mask mode, there is no concept of constant characters. The return value of the component is a string with the user-entered text.	Format	<regexp>true</regexp>	The Minimum and Maximum attribute values are not enforced when you enable the Regular Expression attribute with the value .* in the Format field.

(5 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Filter	Contains a filter (same format as the XMLAPI filter) that allows a user to select an object and add it as an object that is referenced when the script is executed (selection and multiselection fields). For example: <fullClassName>service.Service</fullClassName> <filter> <and> <wildcard name="subscriberPointer" value=”%subscriber:%”/> <or> <equal name="fullClassName" value="apipe.Apipe"/> <equal name="fullClassName" value="epipe.Epipe"/> <equal name="fullClassName" value="fpipe.Fpipe"/> <equal name=”fullClassName" value="vpls.Vpls"/> <equal name="fullClassName" value="mvpls.Mvpls"/> </or> </and> </filter>	Selection Multiselection	<filter></filter>	—
Filter Class Name	Class name of the filter. See the *5620 SAM-O XML Reference* for more information about class names.	Selection Multi-selection	5620 SAM class name	—
List	The items that appear in a list, such as a combo box, selection, or multi-selection component	Combobox	<list> <item> <name/> <uiName/> <value/> </item> <item> <name/> <uiName/> <value/> </item> </list>	—

(6 of 7)

Attribute	Description	Applicable properties	XML tag	Dependencies
Range Set	A range of values, including a minimum and maximum value	Float Long Integer	<range> <rangeSet> <min/> <max/> </rangeSet> <rangeSet> <min/> <max/> </rangeSet> </range>	—
Default Value Converter	Applicable only to XML API configuration templates use for modification only with the Command Type set to Modify. Enables or disables a conversion method that is invoked before an existing network object is being modified using a template. The conversion method is used to set the default values of the Modify Object Using Template form before the form is opened. If the Default Value Converter attribute is disabled, the default values defined in the template are used and the values of the configured network objects are ignored.	All properties	<defaultValueConverter> <enabled></enabled> <body></body> </defaultValueConverter>	—

(7 of 7)

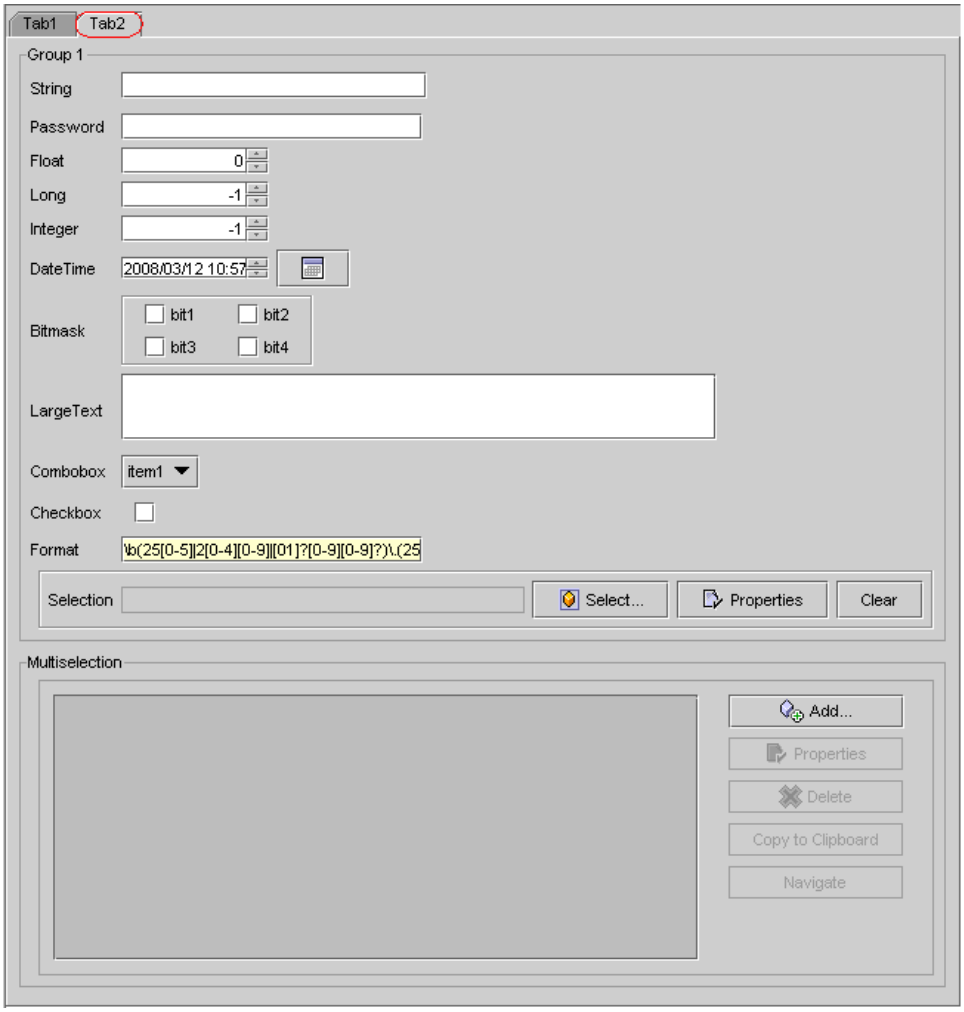
Table 3-7 describes the actions that you can configure using the Action List attribute. Actions are executed depending on the property that the user is changing. For example, if a combo box value is selected, an action can be added that changes another property. All actions are then based on the changed property and can have one or more actions associated to them. If you rename the value for an action list attribute, a warning message appears and the velocity property attribute refreshes with the new value.

Table 3-7 Action List attribute options

Option	Description	Dependencies
Set Action	Identifies that when the filter is satisfied, a value is set on the property.	—
Invoke Action	Allows complex Java calls to be made when the action executes.	—

Figure 3-2 shows an example of all the properties that you can generate using the 5620 SAM GUI Builder.

Figure 3-2 Sample properties on the 5620 SAM GUI Builder



Sample UI configuration using the GUI Builder

The following is a sample configuration to create GRE service tunnels using the 5620 SAM GUI Builder. Procedure 4-6 describes how to access and use the GUI Builder.

Figure 3-3 and 3-4 show the sample UI that you can generate using the GUI Builder. In this sample, two tabs are created—the General tab and the Advanced tab.

Figure 3-3 Sample UI to create service tunnels using the GUI Builder - General tab

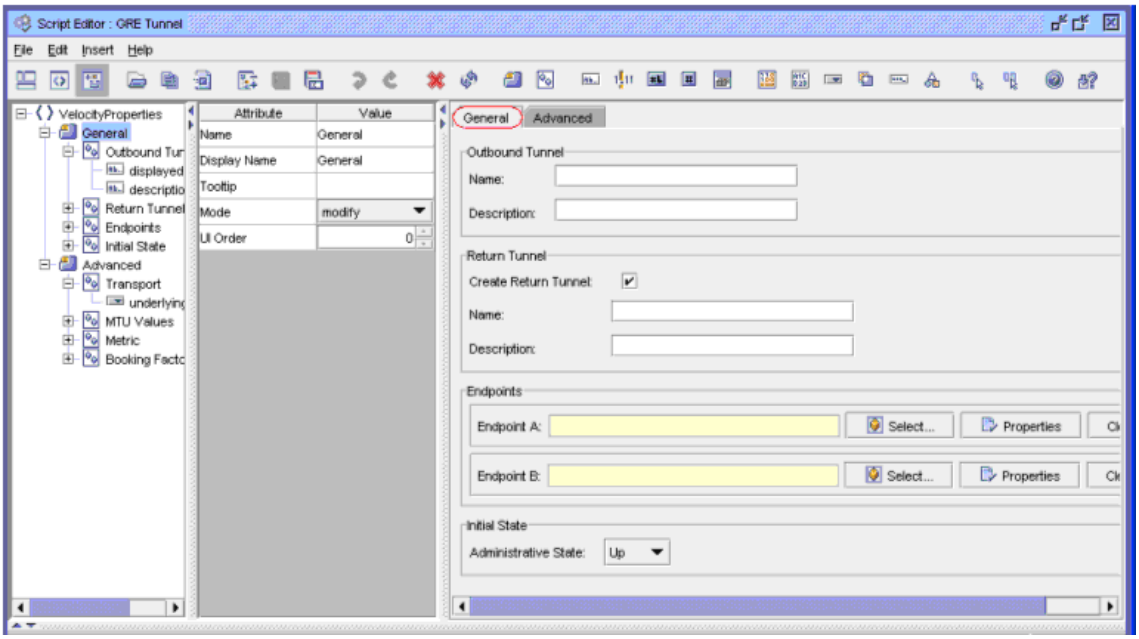


Figure 3-4 Sample UI to create service tunnels using the GUI Builder - Advanced tab

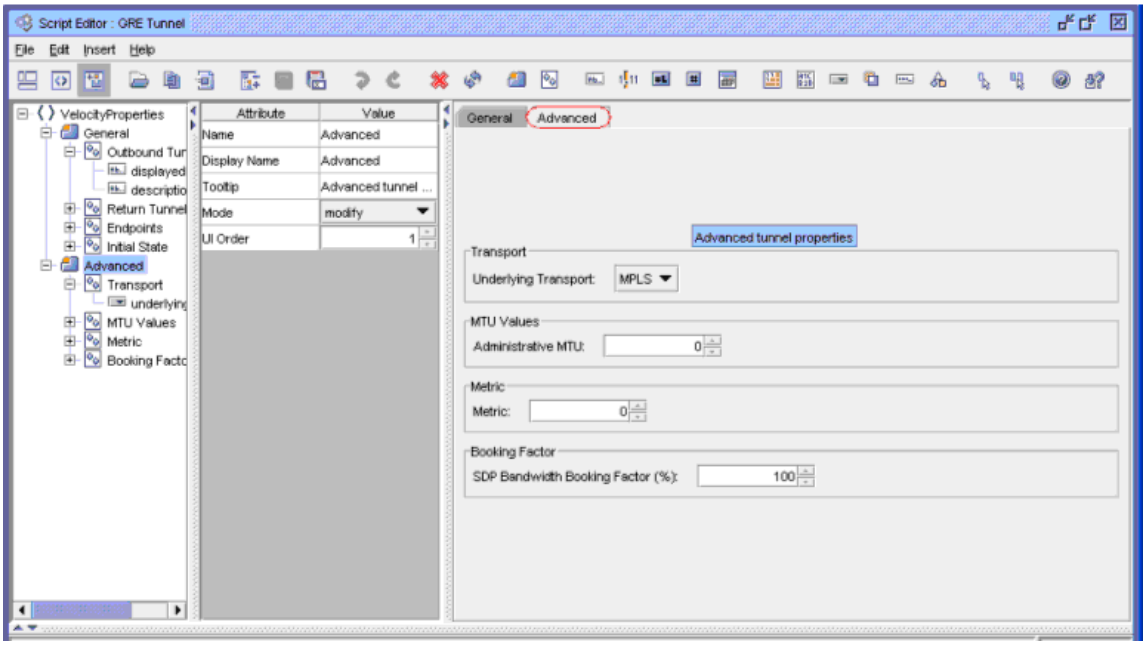


Table 3-8 describes the properties and attributes that are used to create the sample UI in Figures 3-3 and 3-4.

Table 3-8 Sample UI to create service tunnels using the GUI Builder

Properties	Attribute comments	Description
Tab	—	The following tab components are inserted: • General • Advanced See Figures 3-3 and 3-4.
Group	The UI order attribute is set for each group to indicate the order of the group on the GUI. The UI order attribute can be set for all properties.	The following group properties are inserted on the General tab: • Outbound Tunnel (UI Order is 0) • Return Tunnel (UI Order is 1) • Endpoints (UI Order is 2) • Initial State (UI Order is 3) The following group properties are inserted on the Advanced tab: • Transport (UI Order is 0) • MTU Values (UI Order is 1) • Metric (UI Order is 2) • Booking Factor (UI Order is 3) See Figures 3-3 and 3-4.
String	—	The following string properties are inserted on the General tab: • Name • Description See Figure 3-3.
Combobox	The List attribute includes the following list items on the General tab (Initial State combobox): • Up • Down The Default attribute is set to Up. The List attribute includes the following list items on the Advanced tab (Underlying Transport combobox): • MPLS • GRE The Default attribute is set to MPLS.	The Administrative State combobox property is inserted in the Initial State group on the General tab. See Figures 3-6. The Underlying Transport combobox property is inserted in the Transport group on the Advanced tab.
Selection	The Filter Class Name is set to netw.NetworkElement. When you click on the Select button, all the objects configured in the 5620 SAM that belong to the class are listed.	The Endpoint A and Endpoint B selection properties are inserted in the Endpoints group. See Figure 3-7.
Integer	The Range Set attribute is configured with a minimum range item of 576 and a maximum range item of 9194 for the Administrative MTU integer property. The Range Set attribute is configured with a minimum range item of 0 and a maximum range item of 65535 for the Metric integer component. The Range Set attribute is configured with a minimum range item of 0 and a maximum range item of 1000 for the SDP Bandwidth Booking Factor integer component.	The Administrative MTU integer property is inserted in the MTU Values group on the Advanced tab. See Figures 3-8. The Metric integer property is inserted in the Metric group on the Advanced tab. The SDP Bandwidth Booking Factor (%) integer property is inserted in the Booking Factor group on the Advanced tab.

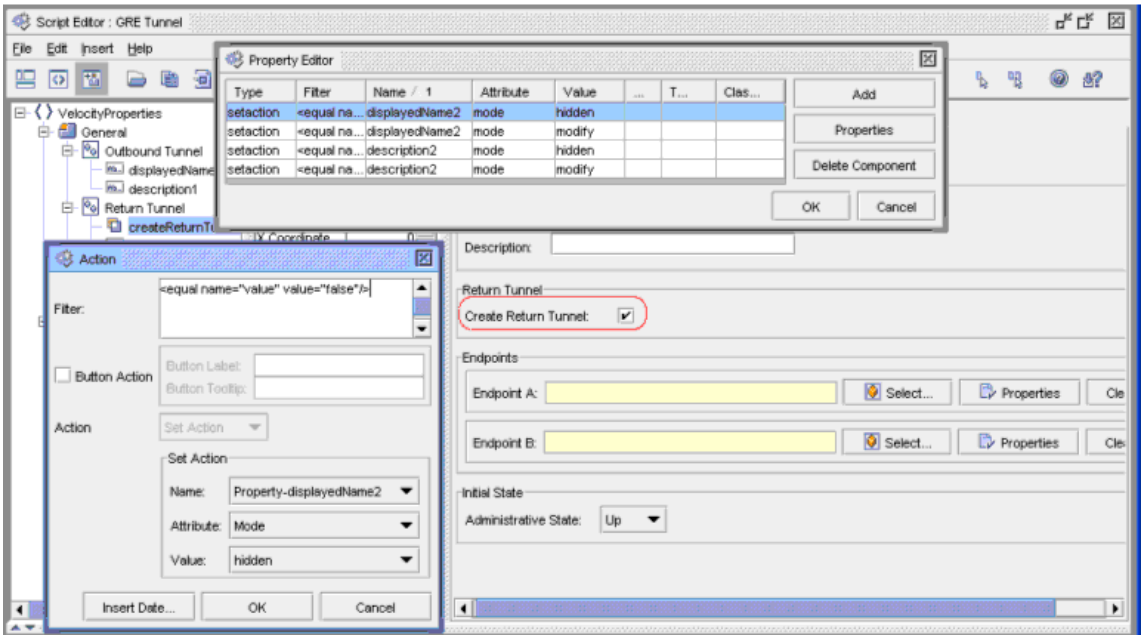
(1 of 2)

Properties	Attribute comments	Description
Checkbox	The Action List attribute is configured to hide the Description and Name string components when the Create Return Tunnel checkbox is unchecked. The default value is checked.	The Create Return Tunnel checkbox property is inserted in the Return Tunnel group on the General tab. See Figures 3-5.

(2 of 2)

Figure 3-5 shows the configuration of an action list. The Action List attribute allows you to specify a filter and the actions to be set or invoked when the specified filter criteria are met. In this sample, the Name and Description properties are hidden when the Create Return Tunnel checkbox property is not selected.

Figure 3-5 Sample UI to create service tunnels using the GUI Builder - Action list



In the Set Action panel on the Action form, the following is configured:

- Name is set to Property-description2
- Attribute is set to Mode
- Value is set to hidden

The Name drop-down menu displays all of the properties that have been selected on the GUI Builder for the form to be created.

The Attribute drop-down menu displays all the attributes that are available for the property that you selected from the Name drop-down menu.

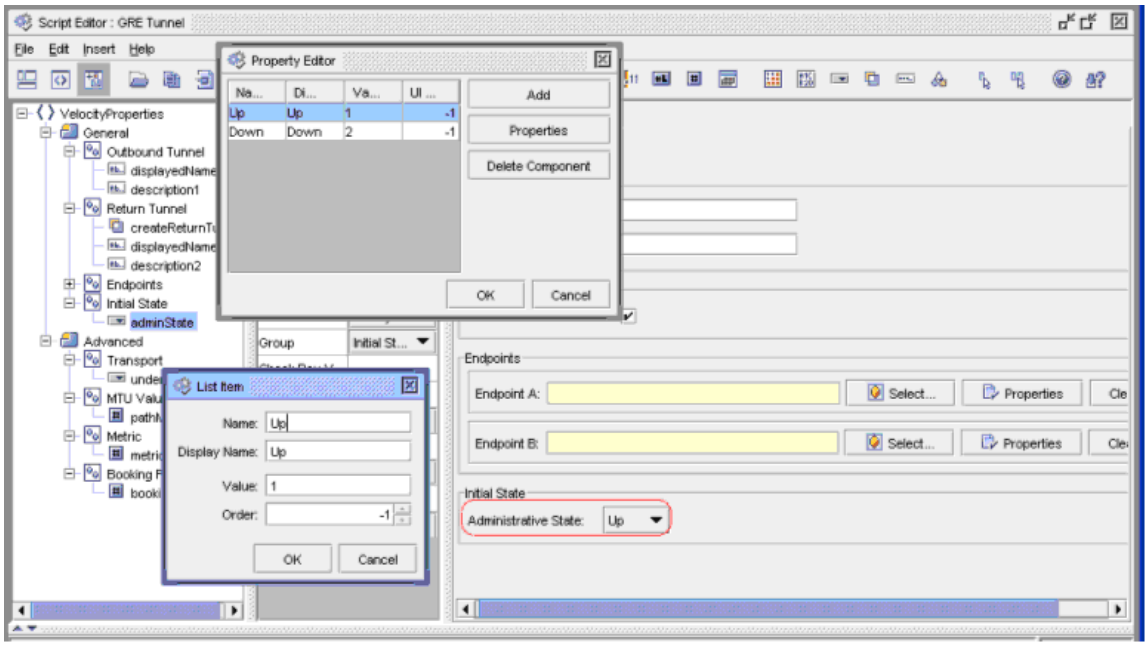
The Value attribute depends on the attribute that you specify in the Attribute drop-down menu. In this sample, Mode is selected from the Attribute drop-down menu, and the available values are modify, mustmodify, hidden, and readonly.

The action—hiding the Description attribute—occurs when the specified filter criteria are met. The following filter is used in this sample:

```
<equal name="value" value="false"/>
```

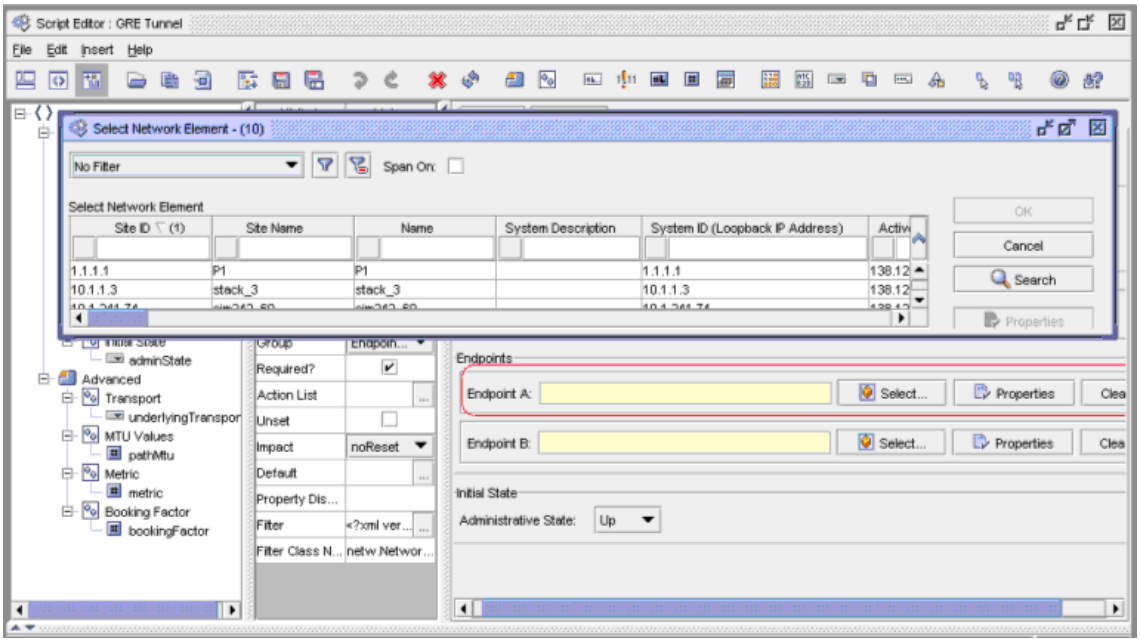
Figure 3-6 shows the configuration of the list items in the Administrative State Combo box, using the List attribute.

Figure 3-6 Sample UI to create service tunnels using the GUI Builder - Combo box list items



When you insert a Selection or Multiselection component, you must specify the class from which the 5620 SAM retrieves the configured objects to create the selection list. In this sample, the netw.NetworkElement class is specified in the Filter Class Name attribute for the Endpoint A and Endpoint B components, as shown in Figure 3-7.

Figure 3-7 Sample UI to create service tunnels using the GUI Builder - Selection property



You can configure a range with a minimum value and maximum value to an Integer component. In this sample, the Range Set attribute is configured for the Administrative MTU property on the Advanced tab, as shown in Figure 3-8.

Figure 3-8 Sample UI to create service tunnels using the GUI Builder - Integer property with range set

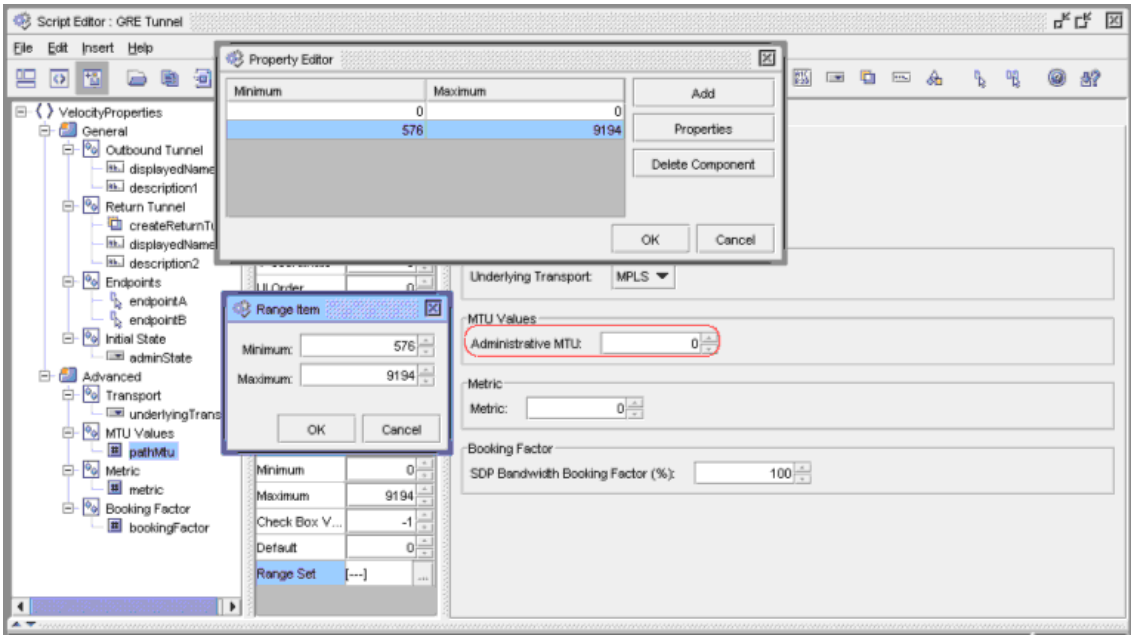
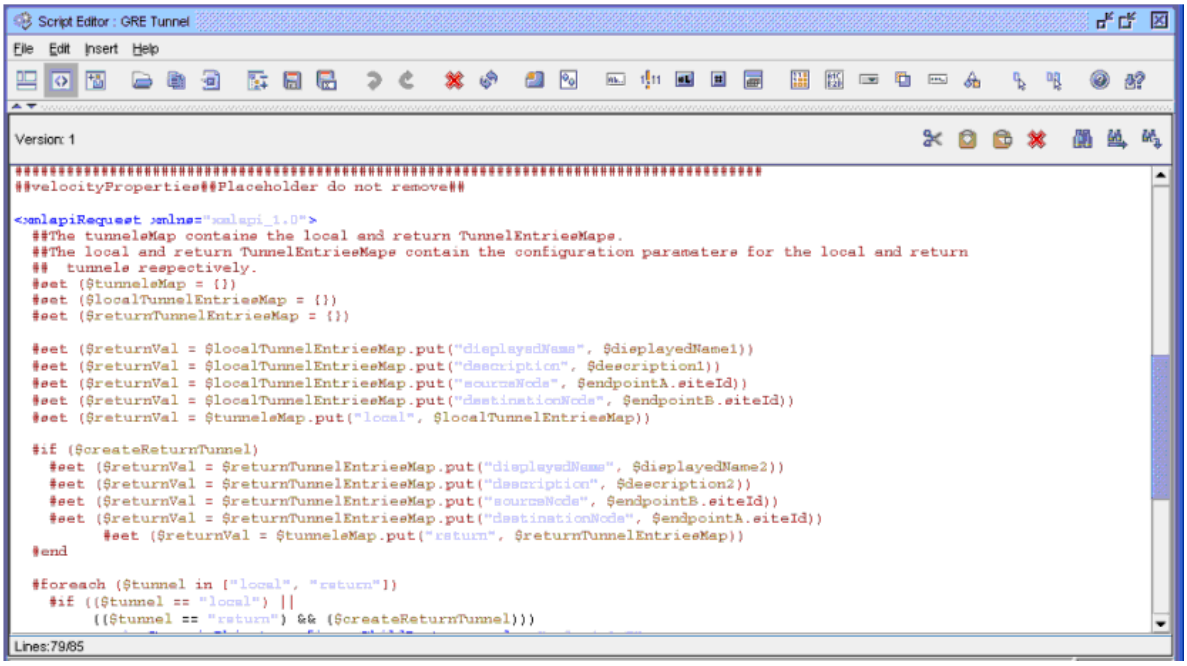


Figure 3-9 shows the Velocity UI header that is generated for the sample configuration.

Figure 3-9 Sample UI to create service tunnels using the GUI Builder - generated Velocity UI header



This sample configuration creates the XML API script, as shown in Code 3-2, to configure the service tunnel between two endpoints. The XML API script is inserted after the Velocity UI header that was automatically generated by the GUI Builder, within the *xmlapiRequest* tag.

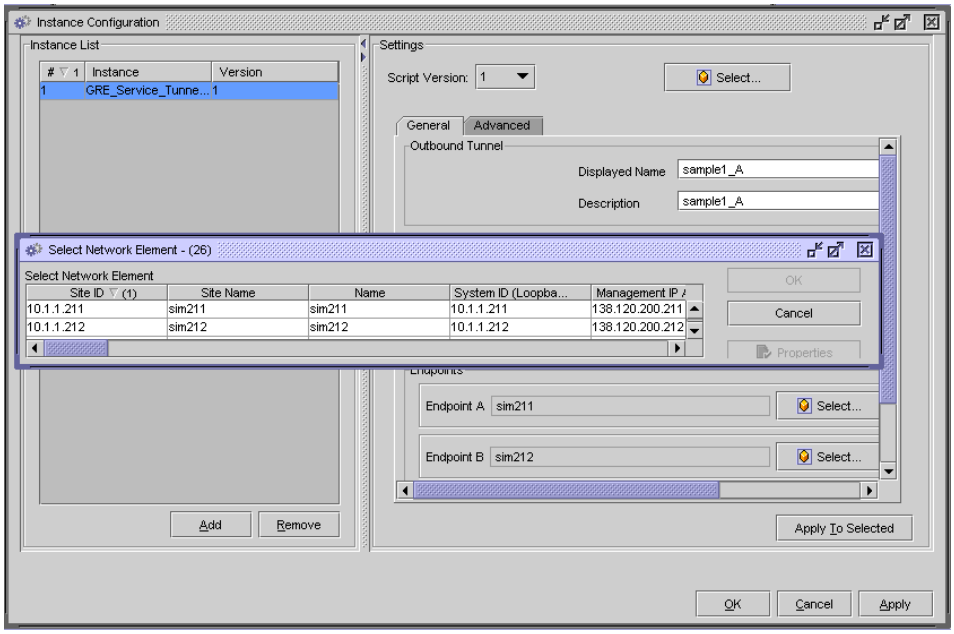
Code 3-2: Sample UI to create service tunnels using the GUI Builder - XML API script

```
#* <velocityProperties>
...
</velocityProperties>
*#

<!-- ****************** Create a-b tunnel ****************** -->
<generic.GenericObject.configureChildInstance xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <synchronousDeploy>true</synchronousDeploy>
    <distinguishedName>serviceTunnel</distinguishedName>
    <childConfigInfo>
      <svt.Tunnel>
        <actionMask><bit>create</bit></actionMask>
        <pathId>0</pathId>
        #if($displayedName1)
          <displayedName>$displayedName1</displayedName>
        #end
        #if($description1)
          <description>$description1</description>
        #end
        <sourceNodeId>#if($endpointA)$endpointA.siteId#end</sourceNo
deId>
        <destinationNodeId>#if($endpointB)$endpointB.siteId#end</des
tinationNodeId>
        <underlyingTransport>gre</underlyingTransport>
```

```
        <administrativeState>$adminState</administrativeState>
        <signallingType>$signalling</signallingType>
        <pathMtu>$pathMtu</pathMtu>
        <advertisedMtuOverride>$advertisedMtuOverride</advertisedMtu
Override>
        <vlanVcEtherType>$vlanVcEtherType</vlanVcEtherType>
        <children-Set/>
      </svt.Tunnel>
    </childConfigInfo>
  </generic.GenericObject.configureChildInstance>

  <!-- ****************** Create b-a tunnel ****************** -->
#if( $createReturnTunnel == "true" )
    <generic.GenericObject.configureChildInstance
xmlns="xmlapi_1.0">
      <deployer>immediate</deployer>
      <synchronousDeploy>true</synchronousDeploy>
      <distinguishedName>serviceTunnel</distinguishedName>
      <childConfigInfo>
        <svt.Tunnel>
          <actionMask><bit>create</bit></actionMask>
          <pathId>0</pathId>
          #if($displayedName2)
            <displayedName>$displayedName2</displayedName>
          #end
          #if($description2)
            <description>$description2</description>
          #end
          <sourceNodeId>#if($endpointB)$endpointB.siteId#end</source
NodeId>
          <destinationNodeId>#if($endpointA)$endpointA.siteId#end</d
estinationNodeId>
          <underlyingTransport>gre</underlyingTransport>
          <administrativeState>$adminState</administrativeState>
          <signallingType>$signalling</signallingType>
          <pathMtu>$pathMtu</pathMtu>
          <advertisedMtuOverride>$advertisedMtuOverride</advertisedM
tuOverride>
          <vlanVcEtherType>$vlanVcEtherType</vlanVcEtherType>
          <children-Set/>
        </svt.Tunnel>
      </childConfigInfo>
    </generic.GenericObject.configureChildInstance>
  #end
```

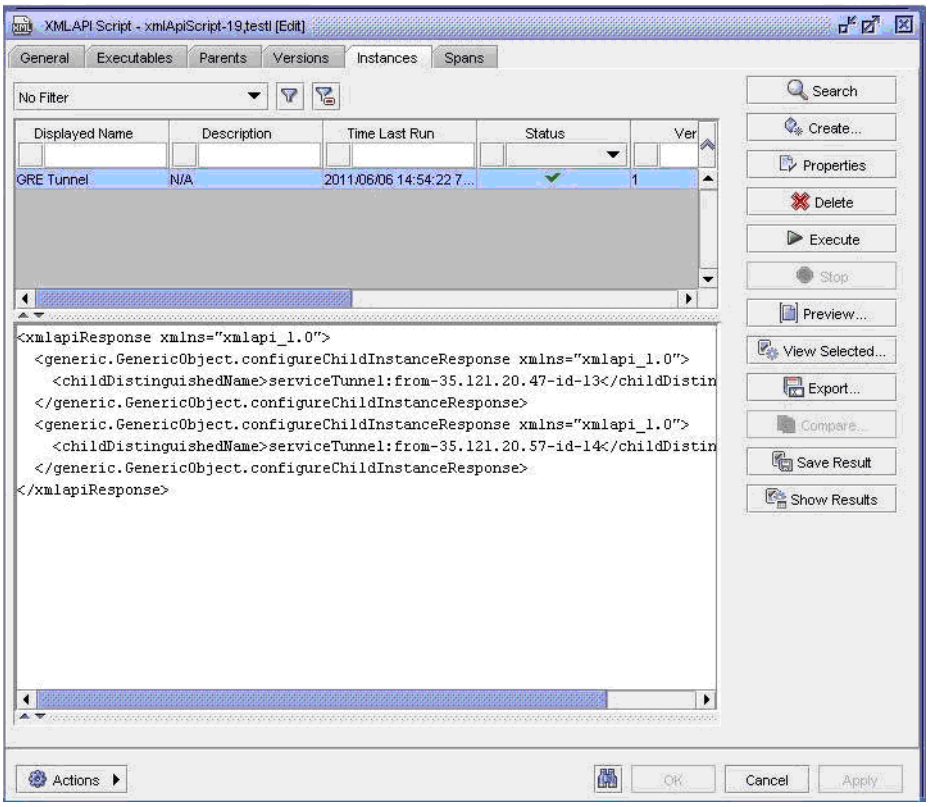
When you save the script and add an instance to the XML API script, as described in Procedure 4-10, the form that you created using the GUI Builder appears on the Instance Configuration form. You can configure the parameters for the instance of the script, as shown in Figure 3-10.

Figure 3-10 Sample UI to create service tunnels using the GUI Builder - Instance Configuration Form



After you apply the configured parameters and save and close the Instance Configuration form, you can execute the script to create the service tunnels, as shown in Figure 3-11. The XML API response displays the result of the configuration. You can then save and show the results.

Figure 3-11 Sample service tunnel creation — instance execution



XML API template property selector

The Property Selector form is an optional form that is only applicable for XML API configuration templates. You can open the form by selecting Edit→Property Selector from the Script Editor menu, or by clicking on the toolbar button. The Property Selector form shows all possible properties for the specified templated object. You can add or remove individual tabs, groups, or properties on the template using the Property Selector form. When a new property is added, the XML API script text is automatically modified and the corresponding XML elements are added or removed. The form has a Product/Version selector that can be used to restrict the properties to a specified network element product and version.

See chapter 6 for more information about XML API templates.

3.7 Velocity Template Language

The Velocity Template Language, or VTL, is a template creation language that can reference Java objects. The 5620SAM Velocity engine allows you to add complex algorithms to CLI scripts, XML API scripts, and templates.

The VTL is used to incorporate dynamic content in an XML API script or a CLI script by using the following types of references to embed dynamic content:

- variables
- properties
- methods

Variables

The short notation of a variable consists of a leading $ character followed by a VTL identifier, which must start with an alphabetic character. The rest of the characters are limited to the following types of characters:

- alphabetic (a to z, or A to Z)
- numeric (0 to 9)
- hyphen (“-”)
- underscore (“_”)

Examples:

```
$ipAddress

$siteId_ptr
```

Properties

The short notation of a property consists of a leading $ character followed by a VTL identifier, followed by a dot character, and another VTL identifier.

Examples:

```
$customer.address

$site.siteId
```

Methods

A method is defined in the Java code and can perform functions, such as performing a calculation or making a decision. Methods are references that consist of a leading $ character, followed by a VTL identifier and a VTL method body. A VTL method body consists of a VTL identifier, followed by a left parenthesis character, followed by an optional parameter list, and followed by a right parenthesis character.

Examples:

```
$vpls.getObjectFullName()

$site.getSiteId()

$alUtil.findByClass("rp.PolicyManager","and(equality(routingInstance
, 1), equality(siteId, $siteId.siteId))")

$policyManagers.iterator()

$policyManagersIterator.next()
```

Comments

Comments allow the inclusion of descriptive text that is not executed by the template engine. A single-line comment begins with ## and finishes at the line end. A multiple-line comment begins with #* and ends with *#. Code 3-3 shows an example of each comment type.

Code 3-3: Comment examples

```
## This is a single-line comment.

#*
This is a multiple-line comment.
These comment lines are not executed by the Velocity engine.
*#
```

Directives

References allow template designers to generate dynamic content for scripts; directives allow the use of script elements to manipulate Java code output, and permit designers to manage the generated XML or CLI scripts.

The following are the most commonly used directives for script and template development:

- set
- if/eslseIf/else
- foreach

A directive line begins with a single # character.

See the Apache Velocity Project online VTL reference for a complete list of the Velocity directives, directive descriptions, and usage examples.

set directive

The set directive assigns a literal value, or the result of an arithmetic or string operation, to a variable. Code 3-4 is an example of set directive usage. See the Apache Velocity Project online VTL reference for a complete list of the supported arithmetic and string functions.

Code 3-4: set directive usage example

```
#set ($detailsValid = "true")
```

if, else, and elseif directives

The if directive adds executive dependencies to a script. If and only if a specified condition is true, the commands between the if directive and the corresponding end statement are executed. Multiple else and elseif directives can be nested in an if directive. The Velocity engine evaluates no expressions after the first expression that is found to be true. Code 3-5 is a usage example for the directives.

Code 3-5: if, else, and elseif directives usage example

```
#set ($detailsValid = "true")
```

```
#set ($qinqEncap = 10)
#set ($qEncap = 2)
#if ($portEncap == $qEncap)
    #if ($innerEncapValue != 0)
      $alUtil.userError("Inner encapsulation should be 0")
    #end
  #elseif ($portEncap == $qinqEncap)
    #if ($innerEncapValue == 0)
      $alUtil.userError("Please specify an inner encap value")
    #end
  #end
```

foreach directive

You can use the foreach directive to create iterative loops. Loops are useful for tasks such as generating multiple lines of output and bulk provisioning. foreach statements can include nested foreach and if/elseif/else statements.

Code 3-6 is an example of foreach directive usage; in the example, the foreach directive loops until it reaches the end of the $saps list variable.

Code 3-6: foreach directive usage example

```
#foreach ($sap in $saps)
  <generic.GenericObject.configureInstance xmlns="xmlapi_1.0">
      <deployer>immediate</deployer>
      <synchronousDeploy>true</synchronousDeploy>
      <distinguishedName>$sap.getObjectFullName()</distinguishedName
      <configInfo>
        <vpls.L2AccessInterface>
          <actionMask>modify</actionMask>
            <ingressPolicyObjectPointer>$policyPointer.getObjectFull
Name()</ingressPolicyObjectPointer>
        </vpls.L2AccessInterface>
      </configInfo>
  </generic.GenericObject.configureInstance>
#end
```

Map

The Java map interface maps keys to values, for example, when a script must perform future data retrieval. For example, you can create a phoneBook map object that has a key named Bob and a value of 212222999. See the Java API for more information.

The map interface example in Code 3-6 shows how to create the interfaceMap object and use the interfaceMap.put method to associate literal values with the displayedName and ip keys. The $ipAddress and $name variables are then set using the interfaceMap.get method to retrieve the key values.

Code 3-7: Sample map

```
#set ($interfaceMap = {} )
#set ($returnVal = $ interfaceMap.put("displayedName", "Interface1"))
       #set ($returnVal = $ interfaceMap.put("ip", "10.1.1.4"))
       #set ($ipAddress = $interfaceMap.get("ip")
## $ipAddress now has a value of 10.1.1.4
```

```
#set ($name = $interfaceMap.get("displayedName")
```

Tools

Velocity tools are utility classes that allow the use of common Velocity helper methods. See Velocity Tools at the Apache Jakarta Project online reference for more information about the Velocity tools API.

Table 3-9 lists the 5620 SAM variable associated with each supported Velocity tool.

Table 3-9 Velocity tools

Velocity tool	Class	5620 SAM variable	Description
AlternatorTool	org.apache.velocity.tools.generic.AlternatorTool	$vmAlternator	Utility class for easily alternating over values in a list; tool for Alternator creation in templates
DateTool	org.apache.velocity.tools.generic.DateTool	$vmDate	Tool for working with Date and Calendar in Velocity templates. Used to: • format and return the current date • format and access Date and Calendar objects • retrieve DateFormat instances • convert to and from various date types
EscapeTool	org.apache.velocity.tools.generic.EscapeTool	$vmEscape	Tool that provides escape-character output methods for Java, JavaScript, HTML, XML, and SQL; provides methods to render VTL characters without using escape characters
IteratorTool	org.apache.velocity.tools.generic.IteratorTool	$vmIterator	Tool for use with foreach loops; wraps a list to let you specify a condition for terminating the loop and reusing the same list in another loop
ListTool	org.apache.velocity.tools.generic.ListTool	$vmList	Tool for working with arrays or lists
MathTool	org.apache.velocity.tools.generic.MathTool	$vmMath	Tool for performing floating-point arithmetic
NumberTool	org.apache.velocity.tools.generic.NumberTool	$vmNumber	Tool for performing numerical operations such as the following: • arbitrary Number object formatting and retrieval • NumberFormat instance retrieval • conversions to and from various number types

(1 of 2)

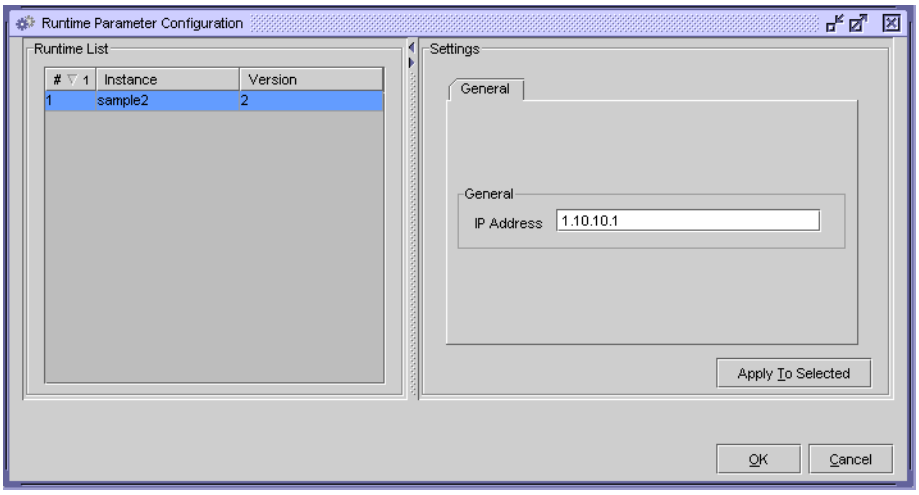
Velocity tool	Class	5620 SAM variable	Description
RenderTool	org.apache.velocity.tools.generic.RenderTool	$vmRender	Tool to expose methods to evaluate the specific strings as VTL (Velocity Template Language) using the current context
SortTool	org.apache.velocity.tools.generic.SortTool	$vmSort	Tool for sorting a collection, such as iterator output or an array, using a set of properties of the objects in the collection
ValueParser	org.apache.velocity.tools.generic.ValueParser	$vmValue	Tool to retrieve and parse String values pulled from a map

(2 of 2)

Runtime parameters

You can include runtime parameters in CLI Velocity and XML API scripts that are configured and applied each time a script is previewed or executed. Runtime parameters are not stored in the configuration of the script. Figure 3-12 shows the runtime parameter configuration form that opens when you preview or execute a script instance that contains runtime parameters.

Figure 3-12 Runtime parameter configuration form



Code 3-8: Sample runtime parameters

```
<property>
    <name>siteid</name>
    <uiName>IP Address:</uiName>
    <type>String</type>
    <default>10.1.1.48</default>
    <tooltip>The IP address of the router</tooltip>
    <required>true</required>
    <runtime>true</runtime>
</property>
```

Callouts

Velocity templates support callouts that allow you to obtain information on the server state, including access to the server object model and information objects. Callouts are evaluated and substituted as the Velocity script is expanded. Callout result content can be passed into other method requests. Method results cannot be expanded in the script.

Callouts can be accessed using the alUtil and alScript parameter information that is passed to the callout.

alUtil parameter

Table 3-10 lists and describes the available callout options for the alUtil parameter.

Table 3-10 alUtil parameter callout options

Command	Description
findByClass(*class_name*, *filter*)	Returns a collection of objects of the *class_name* type that match the *filter* criteria; *filter* is defined as a string that contains one of the following *filterElement* specifications: • empty() • neverPass() • and(*filterElement* [, *filterElement*] [, *filterElement*] …) • or(*filterElement* [, *filterElement*] [, *filterElement*] …) • not(*filterElement*) • quality(*propertyName*, *propertyValue*) • inequality(*propertyName*, *propertyValue*) • less(*propertyName*, *propertyValue*) • lessOrEqual(*propertyName*, *propertyValue*) • greater(*propertyName*, *propertyValue*) • greaterOrEqual(*propertyName*, *propertyValue*) • anyBit(*propertyName*, *propertyValue*) • allBits(*propertyName*, *propertyValue*) • superset(*propertyName*, *propertyValue*) • subset(*propertyName*, *propertyValue*) • wildcard(*propertyName*, *propertyValue*) • between(*propertyName*, *propertyValue*, *propertyValue*) where • *propertyName* is the name of a property in the class being queried • *propertyValue* is the literal value to compare, which is a numeric value, or a string value enclosed in single quotation marks For example: #set($objs = $alUtil.findByClass("file.Policy", "equality(id,6)")) #foreach($obj in $objs) <!-- $obj.getObjectFullName() --> #end

(1 of 2)

Command	Description
generatePattern*(regularExpression)*	Creates a new pattern object for a regular expression; returns the new pattern object, or throws an exception if the expression syntax is invalid A pattern is a string that contains regular-expression match criteria. A pattern can match more than one string; in the following example, the pattern string of a*b matches ab, aab, and aaab. #set ($pattern = $alUtil.generatePattern("a*b")) #set ($matcher = $pattern.matcher("aaab")) #set ($doesMatch = $matcher.matches()) #if ($doesMatch == 'true') #$alUtil.userError("Pattern Match") #end For more examples, open the Browse Examples form from the Script Manager and choose CLI Velocity Script Examples→Configure Basic QoS on 7250.
getProperty(*property*)	Returns one of the following specified 5620 SAM main server properties: • primaryIp—primary IP address • standbyIp— standby IP address; null in a standalone deployment • snmpPort—primary or standalone SNMP port • peerSnmpPort—standby SNMP port; null in a standalone deployment • snmpTrapIpAddress—primary or standalone SNMP trap IP address • peerSnmpTrapIpAddress—standby SNMP trap IP address; null in a standalone deployment
interventionWithAnswer (“*prompt*”, “*response*”)	Interprets a CLI prompt and responds using a specific value; the prompt and response must be on the same CLI line For example, an NE returns the following confirmation prompt after a reboot command: Are you sure you wish to reboot? [y/n] The following statement responds y to the question: reboot $alUtil.interventionWithAnswer (“Are you sure you wish to reboot? [y/n]”, “y”)
interventionWithDefault (“*prompt*”)	Interprets a CLI prompt and responds using the default value; the prompt and response must be on the same CLI line
resolveName(*object_FDN*)	Returns the object specified by *object_FDN*
sleep(*ms*)	Pauses the current script execution for a period of *ms* milliseconds
throw(*userErrorMessage*, *exceptionClass*)	Throws an *exceptionClass* type of exception, and displays *userErrorMessage*
userError(*userErrorMessage*)	Stops the current script execution and displays *userErrorMessage* For example: #if ($customer.isEmpty() $alUtil.userError(.No Customer Set.) #else #set ($subscriberPointer = $customer.getObjectFullName()) #end Command Details/Commands Failed No Customer Set Total:1

(2 of 2)

alScript parameter

Table 3-11 lists and describes the available callout options for the alScript parameter.

Table 3-11 alScript parameter callout options

Command	Description
getCliTarget()	Returns a CLI script target NE object
getTarget()	Returns an XML API script target object

Accessor method get<propertyName>

All object properties have an accessor method that returns the value of the property. The name of the accessor method is get*PropertyName*, where *PropertyName* is the name of the object property, such as objectFullName.

Control script methods

To control the execution of multiple scripts in a custom workflow, the 5620 SAM script cascading function requires a control script. A control script implements the decision logic of a flow and also controls the execution flow. The following tables list and describe the methods associated with each control script variable and function. See section 4.5 for more information about control scripts.

alCtrl methods

Control scripts use the standard Velocity variables, properties, methods and tools described in this guide; control scripts also use the alCtrl variable, which has methods for cascading functions such as the following:

- defining invokers for other scripts and templates
- passing variables to other scripts in the execution flow
- passing control to other scripts in the execution flow
- publishing script execution results

Table 3-12 lists and describes the alCtrl methods.

Table 3-12 alCtrl methods

Method	Description
addResultItem('*text_description*')	Publish a result item without specifying an object FDN; the item is appended in XML format to the control script result object When you execute a script using the 5620 SAM GUI and the script calls this method, each result item is listed in the result table of the script execution window. For example: $alCtrl.addResultItem('Created Service')

(1 of 3)

Method	Description
addResultItem('*text_description*', '*object_FDN*')	Publish a result item and specify the object FDN; the item is appended in XML format to the control script result object When you execute a script using the 5620 SAM GUI and the script calls this method, each result item is listed in the result table of the script execution window. For example: $alCtrl.addResultItem('Created Service', $createdService.getObjectFullName())
addReturnValue('*object_key*', '*object_value*')	Specifies a literal return value to pass to the parent script when the current script exits For example, to pass the name of a newly created service: $alCtrl.addReturnValue('createdService', $createdService)
addReturnValue('*variable_type*', '*variable_name*', '*variable_value*')	Specifies a variable return value to pass to the parent script when the current script exits For example: #set($myVariable = $alCtrl.defineVariable('string', 'myName', 'My Value')) $alCtrl.addReturnValue($myVariable)
defineCliScriptInvoker('*script_name*')	Defines a new invoker instance for a CLI script Returned value type: CliScriptInvoker For example: #set($myInvoker = $alCtrl.defineCliScriptInvoker('My Script Name'))
defineControlScriptInvoker('*script_name*')	Defines a new invoker instance for a control script Returned value type: ControlScriptInvoker For example: #set($myInvoker = $alCtrl.defineControlScriptInvoker('My Script Name'))
defineVariable('*variable_type*', '*variable_name*')	Defines a new variable that can be passed to another script; does not set the variable Returned value type: variable For example: #set($myVariable = $alCtrl.defineVariable('string', 'myName')) $myVariable.setValue('My Value')
defineVariable('*variable_type*', '*variable_name*', '*variable_value*')	Defines a new variable that can be passed to another script, and sets the variable Returned value type: variable For example: #set($myVariable = $alCtrl.defineVariable('string', 'myName', 'My Value'))
defineXmlApiScriptInvoker('*script_name*')	Defines a new invoker instance for an XML API script Returned value type: XmlApiScriptInvoker For example: #set($myInvoker = $alCtrl.defineXmlApiScriptInvoker('My Script Name'))
exitNow()	Immediately stops the current control script execution For example: $alCtrl.exitNow()

(2 of 3)

Method	Description
exitNow($*invoker_name*)	Immediately stops the current control script execution and invokes the specified script; the 5620 SAM GUI displays the script in the tree-based execution view The parameter is an invoker of any script type. For example, to invoke an XML API script: #set($myInvoker = $alCtrl.defineXmlApiScriptInvoker('My Script Name')) $myInvoker.addVariable('string', 'myName', 'My value') $alCtrl.exitNow($myInvoker)

(3 of 3)

Variable methods

Variable methods allow the manipulation of variables in an execution flow. See section 5.5 for the list of supported variable types.

Table 3-13 lists and describes the variable methods.

Table 3-13 Variable methods

Method	Description
addItem('*object_FDN*')	Adds an item to a multiselection variable
clearBit()	Clears a specified bit in a bitmask variable
createCopy()	Creates a copy
isBitmask()	Returns true if the variable is a bitmask Returned value type: boolean
isList()	Returns true if the variable is a multiselection Returned value type: boolean
setBit()	Sets the specified bit in a bitmask variable
setValue()	Sets a variable of any type except multiselection or bitmask

ScriptInvoker methods

You can use ScriptInvoker methods to manage script invoker variables and functions. Table 3-14 lists and describes the ScriptInvoker methods.

Table 3-14 ScriptInvoker methods

Method	Description
addVariable('*type*', '*name*')	Adds a new local variable to the invoker, but does not set the variable Returned value type: variable For example: #set($myVariable = $myInvoker.addVariable(('string', 'myName')) $myVariable.setValue('My Value')

(1 of 3)

Method	Description
addVariable('*type*', '*name*', '*value*')	Adds a new local variable to an invoker, and sets the variable Returned value type: variable For example: #set($myVariable = $myInvoker.addVariable(('string', 'myName', 'My Value'))
addVariable($*variable_name*)	Adds a new copy of a variable to the invoker, and returns a copy of the variable Returned value type: variable For example: #set($myVariableTwo = $myInvoker.addVariable($myVariableOne))
disableHandler()	Disables the automatic invocation of the handler control script that is associated with an XML API script or a CLI script. By default, handler script invocation is enabled. For example: $myInvoker.disableHandler()
enableHandler()	Enables the automatic invocation of the handler control script that is associated with an XML API script or a CLI script. By default, handler script invocation is enabled. For example: $myInvoker.enableHandler()
exportVariable('*type*', '*name*')	Adds a new exported variable to the invoker, but does not set the variable The exported variable is available to all directly or indirectly invoked control scripts in the execution flow. Returned value type: variable For example: #set($myVariable = $myInvoker.exportVariable(('string', 'myName')) $myVariable.setValue('My Value')
exportVariable('*type*', '*name*', '*value*')	Adds a new exported variable to the invoker, and sets the variable The exported variable is available to all directly or indirectly invoked control scripts in the execution flow. Returned value type: variable For example: #set($myVariable = $myInvoker.exportVariable(('string', 'myName', 'My Value'))
exportVariable($*variable_name*)	Adds a new copy of a variable to the invoker, and returns a copy of the variable. The exported variable is available to all directly or indirectly invoked control scripts in the execution flow. Returned value type: variable For example: #set($myVariableTwo = $myInvoker.exportVariable($myVariableOne))
getExportedVariable('*variable_name*')	Returns the value of an exported variable added to this invoker Returned value type: variable
getScriptName()	Returns the script name Returned value type: string
getVariable($*variable_name*)	Returns the value of a local variable added to the invoker Returned value type: variable

(2 of 3)

Method	Description
isHandlerEnabled()	Returns true if handler script invocation is enabled Returned value type: boolean
invokeScript()	Invokes an XML API script or a control script Returned value type: ScriptResult For example: #set($myResult = $myInvoker.invokeScript())
invokeScript('*NE_FDN*')	Invokes a CLI script on a specified NE Returned value type: ScriptResult For example: #set($myResult = $myInvoker.invokeScript('network:23.20.2.4'))

(3 of 3)

ScriptResult nethods

The result of invoking a script is a Java object that is accessible to the invoking control script. Table 3-15 lists and describes the ScriptResult methods.

Table 3-15 ScriptResult methods

Method	Description
get(*index*)	Returns the result object at the position specified by *index*, where *index* is a positive integer The returned object varies by script type. For an XML API script, the object contains the XML API command output. For a CLI script, the object is the name of the result file that contains the CLI command output. Returned value type: object For example: #set($myFirstValue = $myResult.get(0))
getValues()	Returns the list of result objects from an invoked script Returned value type: list For example: #foreach($value in $myResult.values) $value #end
isEmpty()	Returns true if an invoked script returns no results Returned value type: boolean
getFailed()	Returns true if the script execution is failed Returned value type: boolean For example: #if($myResult.failed) do something #end

(1 of 2)

Method	Description
getSucceeded()	Returns true if the script execution is successful Returned value type: boolean For example: #if($myResult.succeeded) do something #end
getErrorMessage()	Returns a non-empty error message when the execution of an invoked script is failed Returned value type: string For example: $myResult.errorMessage
getResultFile()	Returns the ResultFile object that contains the lines of script result output, for example, CLI command output that requires iterative parsing Returned value type: ResultFile
getReturnValue('*variable_name*')	Returns the return value that a child control script specifies using the $alCtrl.addReturnValue method Returned value type: object For example: #set($value = $myInvoker.getReturnValue('variableName'))

(2 of 2)

ResultFile methods

Using the getResultFile method described in Table 3-15, you can return a ResultFile instance. Table 3-16 lists and describes the ResultFile methods.

Table 3-16 ResultFile methods

Method	Description
getSize()	Returns the size, in bytes, of the result file Returned value type: integer

(1 of 2)

Method	Description
lineIterator()	Returns a new iterator that iterates through each line of the file; an iterator can be used only once Alcatel-Lucent recommends that you close each iterator instance individually using the close() method. However, you can call the $resultFile.close() method at any point to ensure that all iterators of a file are closed, if required. Returned value type: ResultLine iterator instance Example 1—To echo each line of the file: #set($lineIterator = $resultFile.lineIterator()) #set($lines = $vmIterator.wrap($lineIterator)) #foreach($line in $lines) $lines.more() #end $lineIterator.close() Example 2—To exit the loop before the final line of the file is read; requires a condition inside the loop to stop the script from calling $lines.more(): #foreach($line in $lines) ## Use the $line variable and determine if more lines ## need to be processed; if not, set the $shouldContinue flag to false #if($shouldContinue) $lines.more() #end #end $lineIterator.close()
lineIterator('*regular_expression*')	Returns a new iterator that iterates each line that matches a regular expression See the lineIterator() entry in this table for more information.

(2 of 2)

ResultLine methods

Using the lineIterator method described in Table 3-16, you can return a ResultLine instance from a ResultFile instance. Table 3-17 lists and describes the ResultLine methods.

Table 3-17 ResultLine methods

Method	Description
getLineNumber()	Returns a line number Returned value type: integer
getLine()	Returns a line Returned value type: string
toString()	Returns a line Returned value type: string

Script and template management

4 — Script management

4.1 Script manager overview

You can use the 5620 SAM GUI to manage CLI scripts for configuring NEs, and XML API scripts for performing complex 5620 SAM tasks. The 5620 SAM Script Manager supports script functions that include the following:

- creation, configuration, and testing
- version control
- organization in logical groups called script bundles
- interoperation between scripts, which is called script cascading
- script scheduled tasks
- control of script execution on managed NEs
- result display, comparison, and storage
- UI creation

See chapter 5 for information about script cascading.

Caution — A script that is not correctly created or applied can seriously damage the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage and ensure that each script is verified and validated before he script t is run on a device in a live network.

Note — In a Script (Create) or Script (Edit) form, an asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

The 5620 SAM user that creates a script is the script owner. The script owner can restrict access to, or lock, the owner-created XML API script instances or CLI script targets. When a script is locked during creation or modification, only the script owner can execute, modify, or delete the instances or targets that they create. Another user can, however, copy a script instance or target, and then execute, modify, or delete the copied item, as required.

If the 5620 SAM user account of a script owner is deleted, or if the script predates 5620 SAM Release 9.0 R3, the script has no owner. The existing script instances or targets cannot be locked, but can be copied by a user, and the copies can then be executed, modified, locked, or deleted by the user.

4.2 Script bundles

A script bundle is a container for scripts and templates, and the dependencies, called bindings, between them. The use of bundles also facilitates the import and export of groups of related scripts, templates, and the bindings between scripts. Using bundles helps to manage functions such as script cascading, which is described in chapter 5.

Bundle creation, modification, and deletion require the Script Management scope of command role.

Bundle members

The following object types can be included as bundle members:

- CLI script, if the script contains Velocity content
- control script
- template
- XML API script

When a bundle is administratively enabled or disabled, the bundle members are also enabled or disabled. Changing the administrative state of a member does not affect the bundle state.

Note — When you delete a bundle, the member scripts are also deleted.

Bundle modes

The following rules apply to switching bundle and member modes:

- Switching a bundle from Released to Draft mode does not affect the mode of the bundle members. However, the bundle controls which version of a member script is the latest Released version.
- Switching a bundle from Draft to Released mode sets the mode of each bundle member to Released; a Released bundle cannot contain members in Draft mode.
- You cannot switch a bundle member to Draft mode when the bundle is in Released mode.

When a bundle is in Draft mode, a 5620 SAM Script Execution operator has access to the latest Released version of each member at the time of the mode switch. This version value is the Use Version value displayed in the Members table on the bundle properties form. A Script Management operator, however, has access to the latest version of each member, regardless of the mode, to facilitate bundle modification and testing.

The Use Version value of each bundle member is 0, which means that version control is not currently in effect, under the following conditions:

- when the bundle is in Draft mode and has not been in Released mode before
- when the bundle is in Draft mode and the member has not previously been in Released mode, so there is no latest Released version of the member
- when the bundle is in Released mode; a Released bundle can contain only members in Released mode, so there are no Draft member versions to monitor

When a bundle is in Draft mode, the Use Version value of each member is the version of the member at the time of the mode switch.

Bundle import and export

You can import and export bundles for distribution between 5620 SAM systems; for example, between a system in a test environment and a system in a live network. The import of a bundle is called a bundle installation. The import of a bundle that has the same name as another bundle is called a bundle upgrade.

During a bundle installation or upgrade, the 5620 SAM checks each script in the bundle to see whether another bundle contains a member with the same name. If two members have the same name, the new bundle is not created and an error is logged.

During a bundle upgrade, the 5620 SAM automatically manages the member versions. If none of the scripts in the original bundle has the same name as an imported script, the 5620 SAM creates a new script and version in the bundle. Otherwise, the 5620 SAM evaluates the script using the following rules.

- If the script content is the same, the version number does not change.
- If the script content is different, one of the following occurs.
  - If the latest version is in Draft mode, the new script replaces the old script, and the version remains the same.
  - If the latest version is in Released mode, a new version is created and set to Draft mode. The Released version is not affected.

A bundle upgrade does not remove a script from a bundle; however, the existing bindings are removed before the new bindings are created from the imported bundle.

The 5620 SAM displays and logs the import and export operations as they are performed. The historical import and export log entries are listed on the bundle properties form.

4.3 CLI scripts

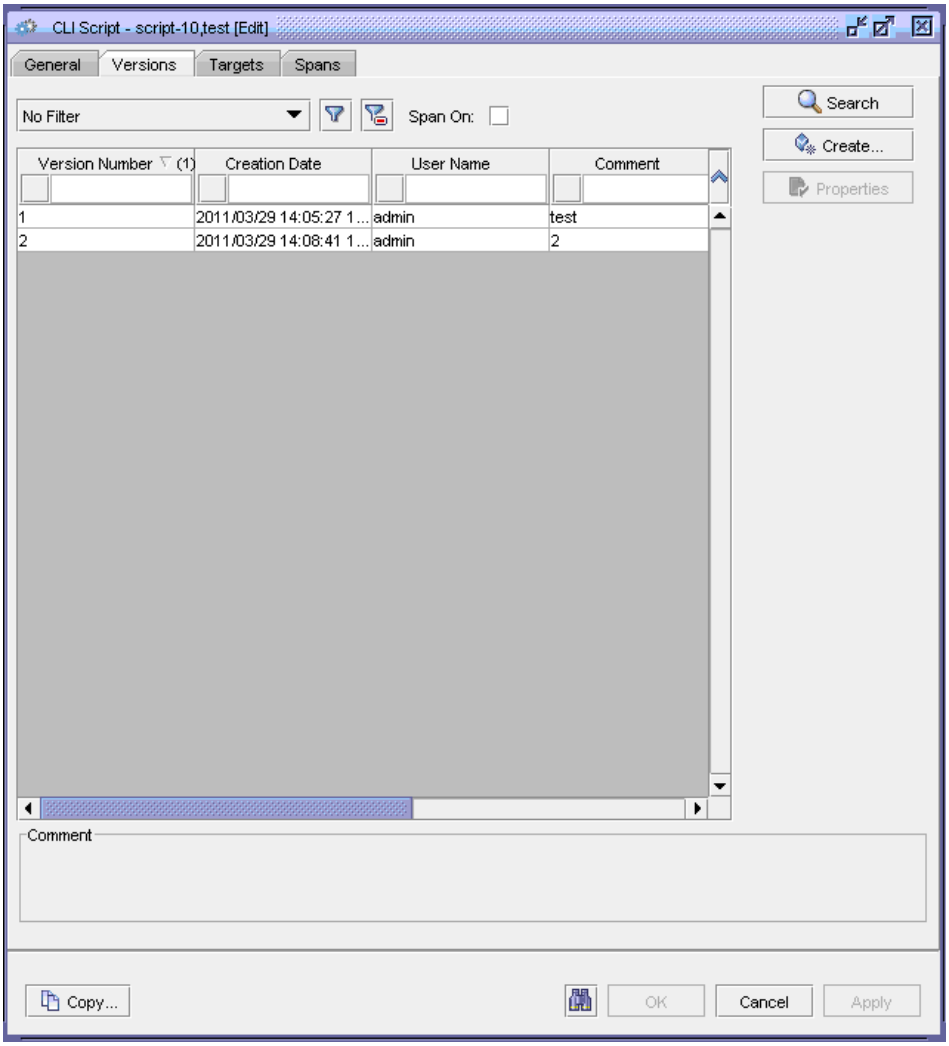
You can create CLI scripts to configure NEs or to display NE information. A CLI script can be executed on single or multiple targets using the 5620 SAM CLI script manager. For example, you can create a script for each type of NE to retrieve statistics information and save the results for further processing. You can also create CLI scripts with configuration commands to configure NEs.

See the appropriate device documentation for the applicable CLI commands, structure, and usage. See section 4.9 for 5620 SAM script management procedures.

Versions

You can use the script editor to add and edit versions of CLI or XML API scripts. Older versions are maintained when you add a new version. When you edit an existing version, the modified script is added as a new version. Figure 4-1 shows a sample CLI script version list.

Figure 4-1 Script version list



The Versions tab displays a list of script versions that are saved in the database. Any changes that are saved in the Script Editor window result in a new version of the script being created.

Note — Sample scripts are available. See Procedure 4-7 for information about how to browse and create sample scripts, and Appendix B for descriptions of script management examples.

Consider the following when you add a version of a CLI or XML API script:

- You can enable the Use Latest Version parameter for a CLI script so that all of the configured targets of a script are associated with the latest version of the script.
- You can enable the Use Latest Version parameter for an XML API script so that all of the XML API instances are associated with the latest version of the script.

CLI script syntax

You can insert parameter tags and intervention tags in a script to specify configurations that are native to the target NE. Table 4-1 describes the supported CLI script syntax.

Table 4-1 CLI script syntax

Tag	Description
#comments	Specifies that the text that follows is treated as comments. Comments are ignored by the 5620 SAM when it generates CLI commands from the script.
$*parameter*$	Specifies a script parameter value substitution. The value of this tag is specified by the user on a per-target object basis.
$*parameter*$(default *abc*) $*parameter*$(default *abc*) (runtime) or $*parameter*$ *abc* (runtime)	Specifies a script parameter value substitution with a default value. The value of this tag is specified by the user on a per-target object basis.
	The runtime tag identifies the parameter as non-persistent. The user must fill in the parameter every time the script is executed.
$question=’’’’answer=’’’’$	Specifies that an answer is expected in response to a command. When the system sends a command that has a question tag after it, it automatically responds with the answer to that question on the NE.
$question=’’’’’$	

Note — The 5620 SAM script manager does not perform a syntax analysis on scripts.

Targets

A target is a managed Alcatel-Lucent or generic NE. Script configuration includes specifying the types of NEs that can be targets, then choosing the target NEs. You can specify multiple NE types as targets, but the 5620 SAM accepts an NE as a target only if the NE type is compatible with the script type. For example, you can create an OmniSwitch script, specify each OmniSwitch model as an NE type for the script, then choose the managed OmniSwitch NEs as script targets.

The Targets tab displays all of the script instances that are invoked when you click on the Execute button on the General tab.

Note 1 — You cannot execute a CLI script if it is not associated with at least one target. Up to 10 CLI scripts can be executed at one time using the script manager.

Note 2 — If a script is locked when a user creates a script target, only that user can subsequently execute, modify, or delete the target.

The user name of the target creator is shown in the Created By User column on the Targets tab of the CLI script properties form.

Consider the following when you add a target to a version of a CLI script:

- You can add a target to a script multiple times and change the name of each entry of the target.
- By adding a target multiple times and enabling the Use Latest Version parameter, you can send the same script version to the same NE several times, each time with a different parameter value.

See Procedure 4-9 for more information about how to add or remove targets.

If the Use Latest Version parameter is enabled for the script, the latest version of the script is executed on all the target entries, with the parameter value that is applied to each.

Secure CLI communication with targets

By default, the 5620 SAM uses Telnet for CLI communication with managed NEs. SSH2 can be used to ensure a secure CLI connection with targets. The mediation policy for the target NEs must be configured with SSH2 and the correct user name, password, and SSH server port. See chapter 14 for more information about configuring SSH2.

Generic NE profiles

You can use the script manager to manage scripts that are executed on managed non-Alcatel-Lucent devices. The generic NE profile that is used for the discovery of generic NEs must be properly configured with CLI read and write access login prompts, and other session information that is specific to the generic NE type.

See chapter 13 in the *5620 SAM User Guide* for more information about configuring generic NE profiles for CLI script management of non-Alcatel-Lucent devices.

4.4 XML API scripts

You can create XML API scripts to display information about objects or to configure objects in the 5620 SAM. XML API script instances allow you to interact with the 5620 SAM through the XML API with OSS commands and by using the Velocity engine.

Note — Only five XML API scripts can be executed at one time in the 5620 SAM.

The 5620 SAM uses Velocity substitution in the script when the script is executed. For more information about the Velocity engine, see section 3.4.

See section 4.9 for information about script procedures.

Versions

The Versions tab displays a list of script versions that are saved in the database. Any changes that are saved in the Script Editor window create a new version of the script.

XML API script syntax

An XML API script can have multiple requests in one script. The multiple requests must be wrapped in a root tag, called *xmlapiRequest*. The *xmlapiRequest* tag appears in the script editor workspace by default:

```
<xmlapiRequest xmlns="xmlapi_1.0">

</xmlapiRequest>
```

Command menu

XML API scripts in the script editor include a contextual menu of available commands. You can insert a 5620 SAM object, attribute, or method in an XML API script by typing the period character (.).

For example, to see the list of available classes, objects, methods, types, and properties in the security package, type the following:

```
<security.
```

A contextual menu appears with a list of objects in the security package. You can choose an item from the list to automatically complete the command, as shown in Figure 4-2.

Figure 4-2 Script editor command menu

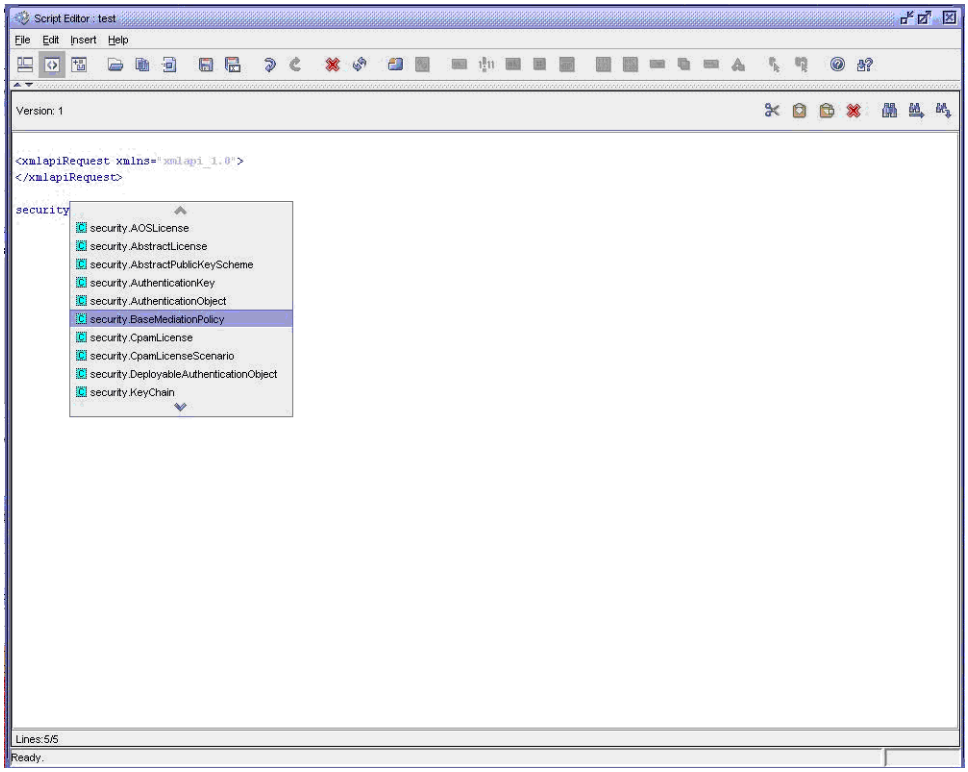


Table 4-2 describes the types of objects that appear in the command menu.

Table 4-2 Command menu options

Type	Description
Class	A class defined in the 5620 SAM When you insert a class, you can open a second menu from which you can choose the properties and methods that belong to the class. If you choose All Properties from the menu, all the properties associated with the class are inserted into the XML script.
Enum	An Enum value defined in the 5620 SAM When you choose Enum, the specific enum value is inserted in the script.
Method	A method from a class When you choose a method, a list of the necessary parameters for the selected method is inserted, in addition to comments and information.
Property/ Overwrite	A property for a class When you choose a property, the tag for the property is inserted, in addition to comments and information about the properties.
Type	A type defined in the 5620 SAM When you insert a type, you can open a second menu from which you can choose the properties and methods that belong to the type.

To exit the command menu, press the Escape key.

Code 4-1 shows a sample XML API script to configure a mediation policy.

Code 4-1: Sample XML API script

```
<generic.GenericObject.configureInstanceWithResult
xmlns="xmlapi_1.0">
    <distinguishedName>pollerManager</distinguishedName>
    <includeChildren>false</includeChildren>
    <deployer>immediate</deployer>
    <configInfo>
        <security.MediationPolicy>
            <actionMask><bit>create</bit></actionMask>
            <displayedName>$name</displayedName><!--
Type:stringMin:0Max:80-->
            <cliPassword>$password</cliPassword> <!-- Type:string-->
            <cliUserName>$username</cliUserName> <!-- Type:string-->
            <community>$community</community> <!--
Type:stringMin:0Max:32-->
            <id>$id</id> <!-- Type:intMin:1Max:65535-->
            <securityModel>snmpv2c</securityModel> <!--
Type:security.SnmpSecurityModel-->
            <snmpRetry>1</snmpRetry> <!-- Type:intMin:0Max:4-->
            <snmpTimeout>10000</snmpTimeout> <!--
Type:intMin:200Max:40000-->
        </security.MediationPolicy>
    </configInfo>
</generic.GenericObject.configureInstanceWithResult>
```

Instances

The Instances tab displays the script instances that run when you click on the Execute button on the General tab. See Procedure 4-10 for information about how to add or remove instances.

Note — If a script is locked when a user creates a script instance, only that user can subsequently execute, modify, or delete the instance.

The user name of the instance creator is shown in the Created By User column on the Instances tab of the XML API script properties form.

4.5 Control scripts

Control scripts are Velocity scripts that are used in script cascading. A control script invokes, or calls, other scripts and templates, passes variables to them, and makes flow decisions based on the results. See chapter 5 for information about script cascading and working with control scripts.

4.6 Script execution

You can execute the scripts that you create in the respective script managers after you have associated targets with CLI scripts, or instances with XML API scripts. The length of time a script takes to execute depends on the overhead; for example, running through a velocity parser, parameter retrieve and generation, result parsing.

Table 4-1 lists and describes the icons that appear when you execute a script to identify the status of the execution.

Note — By default, only status icons are displayed in the 5620 SAM. You can display text values instead of icons by changing the displayIcons=”no” attribute in the “scriptManagement” tag of the nms-client.xml file.

Table 4-3 Script execution status

Status icon	Status	Description
	In Progress	The script manager is attempting to establish a connection with the target and execute the script. Only one script can be executed at a time on a node. You cannot stop scripts that are in progress.
	Queued	The script is queued for execution on the target. You can stop queued scripts.
	Completed	The script has executed successfully.
	Stopped	The script execution has been stopped.
	Communication Error	Invalid CLI user name or password for the target. A connection could not be established.
	Initial	The script has not been executed in the current session.
	Error	The script has executed successfully, but with command failures.

Script scheduled tasks

You can apply a 5620 SAM schedule to a CLI script target or an XML API script instance. A script scheduled task will be automatically executed at the specified time, and optionally repeated at specified intervals. See the *5620 SAM User Guide* for more information on scheduling.

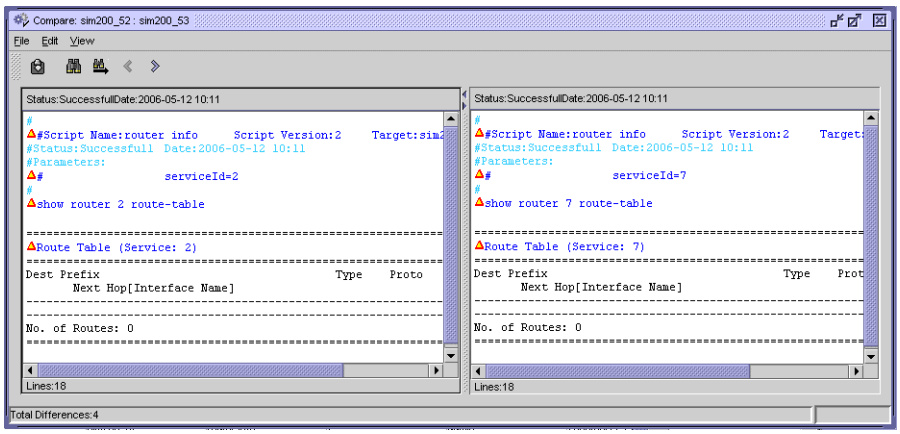
4.7 Script results

You can view the following information for scripts that are executed and for which results have been saved:

- the date of execution
- the user that executed the script
- the version of the script that was executed
- the status of the script execution
- the parameters of the script

You can also view and compare the results of individual and multiple script instances. The number of differences between two script results is displayed. Figure 4-3 shows an example of a comparison of two script executions.

Figure 4-3 Compare form for script results



Note — You can configure the number of results that are stored using a size constraint policy. Results are then deleted based on the execution time, where the oldest results are deleted first. If the policy is not configured for result management, the results are not deleted automatically.

See chapter 43 in the *5620 SAM User Guide* for more information about how to configure a size constraint policy.

4.8 Workflow to use the script manager

1. Ensure that the mediation policy for the device is configured with the correct user name and password for CLI communication (for CLI scripts only).
2. For secure scripting, ensure that SSH2 is properly configured on SSH2-capable devices. Ensure that the mediation policy for the device is configured with SSH2 for CLI communication (for CLI scripts only).
3. To use the script manager with generic NEs, ensure that the generic NE profile is configured with the correct CLI read-write access login prompts and connection information (for CLI scripts only).
4. Configure the general script properties.
5. Create, modify, or import a CLI or XML API script.
6. Use the GUI builder to create a UI and generate a Velocity UI header.
7. For CLI scripts, associate NEs, or targets, with the CLI script or script version, and configure UI parameters.
8. For XML API scripts, add XML API instances and configure UI parameters.

9 If required, create a 5620 SAM schedule and associate it with a CLI script target or an XML API script instance.

10 Preview the script.

11 Execute the script.

12 Review the script execution status.

13 Save or export the script execution results.

14 View the results, or compare results with historical results for the same script. You can also view or compare results for scripts executed on different targets or instances.

4.9 Script management procedures

Use the following procedures to create and manage scripts using the script manager.

Procedure 4-1 To create a CLI script or an XML API script

Perform this procedure to create a CLI script or an XLM API script.

Note — See chapter 5 for information about creating control scripts.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Perform one of the following:

a To create a CLI script, choose CLI Script (Scripting) from the object drop-down menu and click on the Create button. The Script (Create) form opens with the General tab displayed.

b To create an XML API script, choose XMLAPI Script (Scripting) from the object drop-down menu and click on the Create button. The XMLAPI Script (Create) form opens with the General tab displayed.

3 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- Use Latest Version
- Reserve Targets
- Starting Point
- Mode

Enable the Use Latest Version parameter to associate all the targets of the script with the latest version of the CLI script, or to associate the XML API instance with the latest version of the XML API script.

4 If you are configuring a CLI script, configure the following parameters. Otherwise, go to step 6.

- Continue On Command Failure
- Content Type

5 Perform the following steps to specify the script target types.

i Click on the Add button in the NE Types panel. The Select Property - CLI Script form opens.

ii Select one or more NE types in the list and click on the OK button. The Select Property - CLI Script form closes and the NE types are listed on the CLI Script (Create) form.

6 Click on the Apply button.

7 Click on the Versions tab button. Add a version of a CLI or XML API script by performing one of the following steps.

a Create a new version of the script by clicking on the Create button.

b Modify an existing version of the script by choosing the version from the list and clicking on the Properties button.

Note — When you edit a version of a script, it is saved as a new version. The previous version continues to appear in the list of versions.

The Script Editor *script_name* form opens.

8 Create the CLI or XML API script text by perform one of the following steps.

a Import an existing text file with a CLI or XML API script.

- Choose File→Import from the Editor menu. The Import dialog box appears.
- Choose the file to be imported and click on the Import button. The script appears in the script manager editor workspace.
- Modify the script as required.

Note — Sample scripts are available. See Procedure 4-7 for information about how to browse and create sample scripts, and Appendix B for descriptions of script management examples.

b Enter CLI or XML API script text in the script manager editor workspace.

See Procedure 4-4 for more information about using parameter tags in CLI command scripts when you use the script manager editor.

Warning — Scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for CLI and XML API script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

9 Perform one of the following to save the version of the CLI or XML API script:

a Save the script to the script manager.

i Choose File→Save from the Editor menu or click on the Save icon. The Comment dialog box appears.

ii Configure the parameters:

- Network Element Version Information
- Bundle ID

The Network Element Version Information parameter is only configurable if you are creating a CLI script.

iii Click on the OK button.

b Export the script to a local or network text file.

i Choose File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network.

ii Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.

iii Click on the Export button. The script version text file is saved in the specified location.

10 Choose File→Close from the Editor menu. The Script Editor form closes and the Script (Edit) form reappears with the Version tab displayed. The new version of the script appears in the list.

11 If you are creating an XML API script, go to step 14.

12 Associate target NEs with the script.

i Click on the Targets tab button.

ii Click on the Create button. The Target Configuration form opens.

iii Click on the Add button. The Selected Network Elements form opens and lists managed NEs of the types specified in step 5.

iv Select one or more NEs in the list and click on the OK button. The Selected Network Elements form closes and the NEs are listed on the Target Configuration form.

Note — If you choose an NE that is incompatible with the type of script, the 5620 SAM displays an error message and does not commit the changes when you try to close the Target Configuration form.

v Change the displayed name of a target NE, if required, by double-clicking on the Target field and entering the desired name.

vi If you created parameter tags and default values in step 8, apply the default value of each parameter to NEs, as required.

- Choose one or more NEs from the Target List panel.
- Choose the parameter from the Settings panel.
- Change the default value, if required, by double-clicking on the value and entering the default value that you want to apply to the specified targets.
- Click on the Apply To Selected button.

See Procedure 4-4 for more information about using parameter tags in CLI command scripts when you use the script manager editor.

vii If the script includes a Velocity UI header, configure the Velocity parameters in the Settings panel, as required.

viii Click on the OK button. The Target Configuration form closes and the Script (Edit) form reappears with the Targets tab displayed.

13 Go to step 17.

14 Add an instance of a XML API script.

i Click on the Instances tab button.

ii Click on the Create button. The Instance Configuration form opens.

iii Click on the Add button in the Instance List to add an XML API script instance to the list.

iv Change the name of the XML API script instance by double-clicking on the name in the Instance column of the Instance List and entering a name, if required.

v Change the description of the XML API script instance by double-clicking on the text in the Description column of the Instance List and entering new text, if required.

vi If the script includes a Velocity UI header, configure the Velocity parameters, as required.

vii Click on the OK button. The Instance Configuration form closes and the XML API Script (Edit) form reappears with the Instances tab displayed.

15 If the script contains runtime parameters, go to step 16. Otherwise, go to step 17.

16 Perform the following steps to preview the script if it includes runtime parameters.

i Choose a target or instance from the list.

ii Click on the Preview button. The Runtime Parameter Configuration form opens and displays the runtime parameters of the script that is associated with the selected target or instance.

iii Configure the runtime parameters in the Settings panel, as required.

iv Click on the OK button. The Preview form opens and displays a read-only version of the script that is associated with the selected target or instance. Where applicable, the parameters that were inserted in the script are displayed with the configured default values, in addition to the configured runtime parameters.

v To save the script to a local or network text file, perform the following steps, if required.

- Choose File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network.
- Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.
- Click on the Export button. The script version text file is saved in the specified location. The Script (Edit) form reappears.

vi Choose File→Close from the Preview form menu. The Script (Edit) form reappears.

vii Go to step 19.

17 Perform the following steps to preview the script.

i Choose a target or instance from the list.

ii Click on the Preview button. The Preview form opens and displays the version of the CLI or XML API script that is associated with the selected target. Where applicable, the parameters that were inserted in the script are displayed with the configured default values. You cannot edit the script in the Preview form.

iii If required, save the script to a local or network text file by performing the following.

- Choose File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network.
- Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.
- Click on the Export button. The script version text file is saved in the specified location. The Script (Edit) form reappears.

iv Choose File→Close from the Preview form menu. The Script (Edit) form reappears.

18 Click on the Spans tab button. Perform one of the following.

a View the existing span or spans of control applied to the script.

i Choose a span from the list.

ii Click on the Properties button. The Span – *span_name* (View) form opens.

iii Click on the following tab buttons:

- Contents to view all scripts to which the span of control is applied
- Span of Control Profiles to view the span of control profiles that are applied to the span of control
- User Groups to view the user groups with which the span of control profiles are associated
- Users to view the users that belong to the user groups

iv Click on the Cancel button to close the form.

v Go to step 19.

b Apply a span of control to the script.

i Click on the Add button. The Select Span(s) – Script form opens.

ii Select the span or spans of control that you need to add to the script.

iii Click on the OK button. The Select Span(s) – Script form closes and a dialog box appears.

iv Click on the Apply button.

v Click on the OK button to confirm the action.

19 Execute the script.

i Choose one or more targets or instances from the list.

ii Click on the Execute button. If the script contains runtime parameters, go to step iii. Otherwise, go to step 20.

iii The Runtime Parameter Configuration form opens. Configure the runtime parameters in the Settings panel.

iv Click on the OK button. The Runtime Parameter Configuration form closes and the script executes.

Note 1 — The status of the script execution appears in the Status column. The script results appear in the bottom panel of the form.

Note 2 — You can also execute the script on all targets or instances associated with the script by clicking on the Execute button on the General tab. You can view the status and results of the execution on the Targets tab.

20 Perform one of the following to save the results of the script:

a Save the results.

i Click on the Save Result button. A dialog box appears.

ii Click on the OK button.

b Export the results to a local or network text file.

i Click on the Export button. A dialog box appears and prompts you to choose a file storage location on your network.

ii Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.

iii Click on the Export button. The results text file is saved in the specified location.

Procedure 4-2 To modify a CLI script or an XML API script script

Perform this procedure to modify the general parameters of a CLI script or an XML API script.

Note — See chapter 5 for information about modifying control scripts.

Note — If a script is locked when a user creates a script target or instance, only that user can subsequently modify or delete the target or instance.

The user name of the target or instance creator is shown in the Created By User column on the Targets or Instances tab of the script properties form.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

- **a** To modify a CLI script, choose CLI Script (Scripting) from the object drop-down menu.
- **b** To modify an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The Script (Edit) form opens with the General tab displayed.

5 Modify the parameters, as required.

6 Click on the OK button to save the changes and close the form.

Procedure 4-3 To add a script version

Perform this procedure to create a script, modify an existing script, or import a script from a local or network text file.

Warning — Scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

- **a** To add or modify a CLI script version, choose CLI Script (Scripting) from the object drop-down menu.
- **b** To add or modify an XML API script version, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The Script (Edit) form opens with the General tab displayed.

5 Set the Mode parameter to Released.

6 Click on the Versions tab button.

7 Perform one of the following to add a new version of the CLI or XML API script:

a Click on the Create button to open a script editor with a blank workspace.

b Choose a version from the list and click on the Properties button to edit the text of an existing version. The modified text is saved as a new version.

8 Perform one of the following to create or modify the CLI or XML API script:

a Import an existing text file with a CLI or XML API script into the script editor workspace.

- Choose File→Import from the Editor menu. The Import dialog box appears.
- Choose the file to be imported and click on the OK button. The script appears in the script manager editor workspace.
- Modify the script as required.

Note — Sample scripts are available. See Procedure 4-7 for information about how to browse and create sample scripts, and Appendix B for descriptions of script management examples.

b Enter the script in the script manager editor workspace.

See Procedure 4-4 for more information about using parameter tags in CLI scripts when you use the script manager editor.

See section 3.4 for more information about how to configure Velocity in the script manager editor.

9 Save or export the script version, as described in step 9 of Procedure 4-1.

10 Choose File→Close from the Editor menu. The Script Editor form closes and the Script (Edit) form reappears with the Version tab displayed. The new version of the script appears in the list.

Procedure 4-4 To use parameter tags and intervention tags in the CLI script editor

This procedure describes how to use parameter tags in the script editor to create CLI scripts.

Warning — CLI scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for CLI script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Ensure that CLI Script (Scripting) is selected in the object drop-down menu.

3 Perform one of the following:

- **a** To create a new CLI script, perform the following steps:
  - **i** Perform steps 2a to 6 of Procedure 4-1 to configure the general parameters of the CLI script.
  - **ii** Go to step 4.
- **b** Perform the following steps to modify the CLI syntax of an existing CLI script.
  - **i** Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.
  - **ii** Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The Script (Edit) form opens with the General tab displayed.
  - **iii** Go to step 4.

4 Click on the Versions tab button.

5 Click on the Create button to create a new script or double-click on an entry to edit an existing script. The Script Editor form opens.

6 Perform one of the following to create or modify the CLI script text:

a Import an existing text file with a CLI script.

- Choose File→Import from the Editor menu. The Import dialog box appears.
- Choose the file to be imported and click on the OK button. The script appears in the script manager editor workspace.
- Modify the script as required.

b Enter CLI script text in the script manager editor workspace.

Warning — Scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for CLI script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

7 Perform one of the following to insert parameter tags, as required:

a Manually enter the parameter tags in the workspace. The parameter tags are described in Table 4-1.

b Choose Insert→Parameter Tag, or click on the parameter tag icon, and configure the parameters:

- Label
- Default Value

Click on the OK button.

8 Perform one of the following to insert intervention tags, as required:

a Manually enter the intervention tags in the workspace. The intervention tags are described in Table 4-1.

b Choose Insert→Intervention Tag, or click on the intervention tag icon, and configure the parameters:

- Question
- Answer
- Default Value

Click on the OK button.

9 Save or export the script version, as described in step 9 of Procedure 4-1.

10 Choose File→ Close from the Editor menu. The Script Editor form closes and the Script (Edit) form reappears with the Version tab displayed. The new version of the script appears in the list.

Procedure 4-5 To configure UI parameters in Velocity scripts

Perform this procedure to configure the UI parameters that are specified in the Velocity header of a script.

Warning — CLI or XML API scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

Note — By default, the script editor view does not allow Velocity properties to be modified because the properties do not appear in the code view.

1 Choose Application→User Preferences from the 5620 SAM main menu. The Script Manager list form opens.

2 On the Script Management tab, deselect GUI Builder and click on the Apply button or the OK button.

Note — Changes do not affect open script editor windows. You must close and reopen the windows to view the changes.

3 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

4 Perform one of the following:

   a To configure the Velocity header of a CLI script, choose CLI Script (Scripting) from the object drop-down menu.

   b To configure the Velocity header of an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

5 Perform one of the following:

a Perform the following steps to create a new script.

i Click on the Create button to create a new script. The *Script_type* Script (Create) form opens with the General tab displayed.

ii Configure the parameters, as required.

Note — If you are creating a CLI script, you must set the Content Type parameter to Velocity.

b Perform the following steps to open an existing script.

i Configure the list filter criteria to modify an existing script. A list of scripts appears at the bottom of the Script Manager form.

ii Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

6 Click on the Versions tab button.

7 Click on the Create button to create a new script or double-click on an entry to edit an existing script. The Script Editor form opens.

8 Enter the Velocity script in the script editor workspace. See Table 3-5 for more information about the Velocity tags that you can use to generate a UI.

9 Enter the CLI commands or XML API script in the script editor workspace.

10 Save or export the script version, as described in steps 9 and 10 of Procedure 4-1.

11 Perform one of the following:

a If you are configuring a CLI script, click on the Targets tab button.

b If you are configuring an XML API script, click on the Instances tab button.

12 Perform one of the following:

a Perform step 12 of Procedure 4-1 to add a target to a CLI script.

b Perform step 14 of Procedure 4-1 to add an instance to an XML API script.

13 Configure the parameters that are generated by the Velocity UI header in the Settings panel, as required.

14 Execute the script, as described in step 6 of Procedure 4-14.

15 Click on the OK button to close the *Script_type* Script (Edit) form.

Procedure 4-6 To generate a UI using the GUI builder

Warning — CLI or XML API scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

- **a** To configure the Velocity header of a CLI script, choose CLI Script (Scripting) from the object drop-down menu.
- **b** To configure the Velocity header of an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

3 Perform one of the following:

- **a** Create a new script:
  - **i** Click on the Create button to create a script. The *Script_type* Script (Create) form opens with the General tab displayed.
  - **ii** Configure the parameters, as required.

Note — If you are creating a CLI script, you must set the Content Type parameter to Velocity.

- **b** Open an existing script:
  - **i** Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.
  - **ii** Double-click on an entry in the list of scripts, or choose an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

4 Click on the Versions tab button.

5 Click on the Create button to create a script or double-click on a script to modify an existing script. The Script Editor form opens with the GUI Builder and Code view.

Note — See Tables 3-5 and 3-6 for a description of the Velocity properties, property attributes, and dependencies on the GUI Builder form.

6 To insert a tab component, perform the following steps.

i Click on the Insert a tab component button or choose Insert→Tab from the 5620 SAM GUI Builder main menu. The tab component appears in the components tree.

ii Configure the following tab component attributes:

- Name
- Display Name
- Tooltip
- UI order

iii Repeat steps i and ii to add additional tab buttons, as required. Otherwise, go to step 7.

7 To insert a group component, perform the following steps.

i Click on the Insert a group button or choose Insert→Group from the 5620 SAM GUI Builder main menu.

ii Configure the following group component attributes:

- Name
- Display Name
- Tooltip
- UI Order
- Tab
- X Coordinate
- Y Coordinate

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Repeat steps 6 or steps 7 i and ii to add additional tab buttons or groups, as required. Otherwise, go to step 8.

8 To insert a string component, perform the following steps. Otherwise, go to step 9.

i Click on the Insert a string component button or choose Insert→String from the 5620 SAM GUI Builder main menu.

ii Configure the following string component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Default
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

9 To insert a float component, perform the following steps. Otherwise, go to step 10.

i Click on the Insert a float component button or choose Insert→Float from the 5620 SAM GUI Builder main menu.

ii Configure the following float component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Runtime?
- Default
- Minimum
- Maximum
- Checkbox Value

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Range Set attribute. The Property Editor form opens.

ix Click on the Add button. The Range Item form opens.

x Configure the attributes:

- Minimum
- Maximum

xi Click on the OK button. The Range Item form closes.

xii Close the Property Editor form.

10 To insert a long component, perform the following steps. Otherwise, go to step 11.

i Click on the Insert a long component button or choose Insert→Long from the 5620 SAM GUI Builder main menu.

ii Configure the following long component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum
- Default

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Range Set attribute. The Property Editor form opens.

ix Click on the Add button. The Range Item form opens.

x Configure the attributes:

- Minimum
- Maximum

xi Click on the OK button. The Range Item form closes.

xii Close the Property Editor form.

11 To insert an integer component, perform the following steps. Otherwise, go to step 12.

i Click on the Insert an integer component button or choose Insert→Integer from the 5620 SAM GUI Builder main menu.

ii Configure the following integer component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum
- Default

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Range Set attribute. The Property Editor form opens.

ix Click on the Add button. The Range Item form opens.

x Configure the attributes:

- Minimum
- Maximum

xi Click on the OK button. The Range Item form closes.

xii Close the Property Editor form.

12 To insert a date/time component, perform the following steps. Otherwise, go to step 13.

i Click on the Insert a date/time component button or choose Insert→Date from the 5620 SAM GUI Builder main menu.

ii Configure the following date/time component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value - UI Name
- Runtime?
- Action List
- Unset
- Impact
- Minimum
- Maximum
- Default
- Check Box Value
- Use Current Time

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

13 To insert a bit mask component, perform the following steps. Otherwise, go to step 14.

i Click on the Insert a bit mask component button.

ii Configure the following bit mask component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Default
- Required?
- Runtime?

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Mask attribute. The Property Editor form opens.

ix Click on the Add button. The Bitmask form opens.

x Configure the attributes:

- Name
- Display Name
- Default Value

xi Click on the OK button. The Bitmask form closes.

xii Close the Property Editor form.

14 To insert a large text component, perform the following steps. Otherwise, go to step 15.

i Click on the Insert a large text component button or choose Insert→Large Text from the 5620 SAM GUI Builder main menu.

ii Configure the following large text component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum
- Default

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

15 To insert a combo box component, perform the following steps. Otherwise, go to step 16.

i Click on the Insert a combobox component button or choose Insert→Combobox from the 5620 SAM GUI Builder main menu.

ii Configure the following combo box component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum
- Default

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the List attribute to configure the list components. The Property Editor form opens.

ix Click on the Add button. The List Item form opens.

x Configure the attributes:

- Name
- Displayed name
- Value
- Order

xi Click on the OK button. The List Item form closes.

xii Close the Property Editor form.

16 To insert a check box component, perform the following steps. Otherwise, go to step 17.

i Click on the Insert a check box component button or choose Insert→Checkbox from the 5620 SAM GUI Builder main menu.

ii Configure the following check box component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Runtime?
- Default

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

17 To insert a password component, perform the following steps. Otherwise, go to step 18.

i Click on the Insert a password component button or choose Insert→Password from the 5620 SAM GUI Builder main menu.

ii Configure the following password component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Default
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Minimum
- Maximum

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

18 Insert a format component, if necessary. Perform the following steps.

i Click on the Insert a format component button or choose Insert→Format from the 5620 SAM GUI Builder main menu.

ii Configure the following format component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Check Box Value – UI Name
- Check Box Value
- Required?
- Runtime?
- Maximum
- Regular Expression
- Format
- Default

Note 1 — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

Note 2 — The Maximum value is used only for display purposes.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

19 To insert a selection component, perform the following steps. Otherwise, go to step 20.

i Click on the Insert a selection component button or choose Insert→Selection from the 5620 SAM GUI Builder main menu.

ii Configure the following selection component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Required?
- Runtime?
- Property Display Name
- Filter Class Name

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Filter attribute. The Property Editor form opens.

ix Configure the Filter attribute, if required.

x Click on the OK button. The Property Editor form closes.

xi Click on the button beside the Default attribute. The Select *Class_name* list form opens with a list of configured objects for the class identified by the Filter Class Name attribute.

Note 1 — You must configure the Filter Class Name attribute before you can configure the Default attribute.

Note 2 — The object that you choose from the Select *Class_name* list form becomes the default value for the Selection property.

Note 3 — You can remove the default value by clicking on the Clear button on the Select *Class_name* list form.

xii Choose an object from the list and click on the OK button. The Select *Class_name* list form closes.

20 To insert a multi-selection component, perform the following steps. Otherwise, go to step 14.

i Click on the Insert a multi-selection component button or choose Insert→Multi-Selection from the 5620 SAM GUI Builder main menu.

ii Configure the following multi-selection component attributes:

- Name
- Display Name
- Tooltip
- X Coordinate
- Y Coordinate
- UI Order
- Mode
- Group
- Required?
- Runtime?
- Filter Class Name

Note — If you configure the X Coordinate and the Y Coordinate attributes, you must set the UI Order property to –1.

iii Click on the button beside the Action List attribute. The Property Editor form opens.

iv Click on the Add button. The Action form opens.

v Configure the attributes:

- Filter
- Button Action
- Button Label
- Button Tooltip
- Action
- Name
- Attribute
- Value

vi Click on the OK button. The Action form closes.

vii Close the Property Editor form.

viii Click on the button beside the Filter attribute. The Property Editor form opens.

ix Configure the Filter attribute, if required.

x Click on the OK button. The Property Editor form closes.

xi Choose an object from the list and click on the OK button. The Select *Class_name* list form closes.

21 You can delete a property by performing the following steps:

i Click on the property that you need to delete.

ii Click on the Delete button. A dialog box appears.

iii Click on the Yes button to confirm the action.

22 Click on the Save button.

23 Choose File→Close from the 5620 SAM GUI Builder main menu to close the form.

24 Save or export the script version, as described in step 9 of Procedure 4-1.

25 Choose File→Close from the Editor menu. The Script Editor form closes and the Script (Edit) form reappears with the Version tab displayed. The new version of the script appears in the list.

26 Add an instance of a XML API script.

i Click on the Instances tab button.

ii Click on the Create button. The Instance Configuration form opens.

iii Click on the Add button in the Instance List to add an XML API script instance to the list. Repeat this step to add more instances. Although there is no limit to the number of instances you can add to a script, you can execute only five instances at one time.

iv Change the name of the XML API script instance by double-clicking on the name in the Instance column of the Instance List and entering a name, if required.

v Change the description of the XML API script instance by double-clicking on the text in the Description column of the Instance List and entering new text, if required.

vi If the script includes a Velocity UI header, configure the Velocity parameters, as required.

27 Configure the generated UI:

i Press the CTRL or Shift key and choose the instances to which you need to copy the values.

ii Configure the Version Number parameter.

iii Configure the parameters in the Settings panel, as required.

iv Click on the Apply To Selected button to apply the parameters to the selected instances.

28 Perform one of the following to remove one or more instances from the script:

a Choose one or more instances from the list on the Instance Configuration form and click on the Remove button, as required.

b Choose one or more targets from the list on the Instances tab of the XML API Script (Edit) form, and click on the Delete button.

29 Click on the OK button. The Instance Configuration form closes and the XML API Script (Edit) form reappears.

30 Click on the OK button to close the XML API Script (Edit) form.

Procedure 4-7 To browse and create from sample scripts, templates, and script bundles

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Click on the Browse Examples button. The Browse Examples of Scripts form opens.

3 Locate an example script, template, or script bundle object in the tree view.

4 Select the object.

5 If the object is a script bundle, perform the folowing steps.

Note — You can view the description and contents of a bundle member by selecting the member and performing steps 6 and 7.

i Click on the Create Bundle button. The Script Bundle (Crate from example) form opens.

ii Perform Procedure 4-16, beginning at the first step after the opening of the creation form; you must end with the step that closes the creation form.

iii Go to step 10.

6 Click on the Description tab button to view a description of the example.

7 Click on the Preview tab button to view the sample script or template.

8 Click on the Create Script button to create an instance of a sample script or template. The *object* (Create from example) form opens, where *object* is Script or Template.

9 Perform the apprpriate script or template creation procedure, beginning at the first step after the opening of the creation form; you must end with the step that closes the creation form.

10 Click on the Close button. The Browse Examples of Scripts form closes.

11 Close the Script Manager form.

Procedure 4-8 To open and compare script versions

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

   a To open and compare a CLI script version, choose CLI Script (Scripting) from the object drop-down menu.

   b To open and compare an XML API script version, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

5 Click on the Versions tab button.

6 Choose a script from the list, and click on the Properties button. The Script Editor form opens.

7 Choose File→Open Comparison from the Editor menu, or click on the Compare Versions icon to open another version of the script. This version is displayed in a panel beside the version currently being edited.

8 Scroll through the differences between the two scripts by choosing View→Next or View→Previous from the Editor menu, or by clicking on the Next or Previous icon.

9 Choose File→Close Comparison.

10 Click on the OK button to close the script manager.

Procedure 4-9 To add or remove targets

Perform this procedure to add new targets to the list of targets that can be associated with a CLI script.

Note — If a script is locked when a user creates a script target, only that user can subsequently remove the target.

The user name of the target creator is shown in the Created By User column on the Targets tab of the CLI script properties form.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Ensure that CLI Script (Scripting) is selected in the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The Script (Edit) form opens with the General tab displayed.

5 Click on the Targets tab button.

6 Click on the Create button. The Target Configuration form opens.

7 Click on the Add button. The Selected Network Elements form opens and lists managed NEs of the types specified in the NE Types list on the General tab of the Script (Edit) form.

8 Select one or more NEs in the list and click on the OK button. The Selected Network Elements form closes and the NEs are listed on the Target Configuration form.

Note — If you choose an NE that is incompatible with the script, the 5620 SAM displays an error message and does not commit the changes when you try to close the Target Configuration form.

9 Change the displayed name of a target NE, if required, by double-clicking on the Target field and entering the desired name.

10 If parameter tags or intervention tags are used in the script version, apply the appropriate value of the parameters to the NEs, if required.

- **i** Choose one or more NEs from the Target List panel.
- **ii** Choose the parameter from the Settings panel.
- **iii** Change the default value, if required, by double-clicking on the value and entering the value that you want to apply to the specified targets.
- **iv** Click on the Apply To Selected button.

11 Perform the following steps to apply the values of one NE to other NEs in the list.

i From the Target List panel, click on the NE whose values you want to copy. The selected NE is highlighted in blue.

ii Press the CTRL or Shift key and choose the NEs that you want to copy the values to.

iii Click on the Apply To Selected button.

12 Perform one of the following to remove one or more targets from the script:

a Choose one or more targets from the list on the Target Configuration form and click on the Delete button, as required.

b Choose one or more targets from the list on the Targets tab of the Script (Edit) form, and click on the Remove button.

13 Click on the OK button. The Target Configuration form closes and the Script (Edit) form reappears.

14 Click on the OK button to close the Script (Edit) form.

Procedure 4-10 To add or remove instances

Perform this procedure to add new targets to the list of targets that can be associated with a CLI script.

Note — If a script is locked when a user creates a script instance, only that user can subsequently remove the instance.

The user name of the instance creator is shown in the Created By User column on the Instances tab of the XML API script properties form.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Ensure that XML API (Scripting) is selected in the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The XML API Script (Edit) form opens with the General tab displayed.

5 Click on the Instances tab button.

6 Click on the Create button. The Instance Configuration form opens.

7 Click on the Add button in the instance list to add an instance. An instance of the XML API script appears in the list. Repeat this step to add more instances. Although there is no limit to the number of instances you can add to a script, you can execute only five instances at one time.

8 Change the name or description of the XML API script instance, if required.

i Change the name of the XML API script instance by double-clicking on the name in the Instance column of the Instance List and entering a new name.

ii Change the description of the XML API script instance by double-clicking on the text in the Description column of the Instance List and entering new text.

9 If the script includes a Velocity UI header, perform the following steps.

i Press the CTRL or Shift key and choose the instances that you want to copy the values to.

ii Configure the Version Number parameter.

iii Configure the parameters in the Settings panel, as required.

iv Click on the Apply To Selected button to apply the parameters to the selected instance or instances.

10 Perform one of the following to remove one or more instances from the script:

a Choose one or more instances from the list on the Instance Configuration form and click on the Remove button, as required.

b Choose one or more targets from the list on the Instances tab of the XML API Script (Edit) form, and click on the Delete button.

11 Click on the OK button. The Instance Configuration form closes and the XML API Script (Edit) form reappears.

12 Click on the OK button to close the XML API Script (Edit) form.

Procedure 4-11 To copy a script

Perform this procedure to copy the general parameters and the latest version of a script to a new script. Target configuration is not copied to the new script.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

a To copy a CLI script, choose CLI Script (Scripting) from the object drop-down menu.

b To copy an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The Script (Edit) form opens with the General tab displayed.

5 Click on the Copy button. The Select a Version list form opens.

6 Select a script version to copy and click on the OK button. The Select a Version list form closes and the Script (Create) form opens with the General tab displayed.

7 Configure the parameters, as required:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- Starting Point
- Use Latest Version
- Continue On Command Failure
- Mode

Note 1 — You must configure the Name parameter for the copied script.

Note 2 — The Continue On Command Failure parameter is configurable only for CLI scripts.

8 Click on the Versions tab to view the copied script version. See steps 7 to 10 of Procedure 4-3 for more information about how to add a script version.

9 If you are configuring a copy of an XML API script, go to step 13.

10 Perform the following steps to specify the script target types.

i Click on the Add button in the NE Types panel. The Select Property - CLI Script form opens.

ii Select one or more NE types in the list and click on the OK button. The Select Property - CLI Script form closes and the NE types are listed on the CLI Script (Create) form.

11 Associate target NEs with the script.

i Click on the Targets tab button.

ii Click on the Create button. The Target Configuration form opens.

iii Click on the Add button. The Selected Network Elements form opens and lists managed NEs of the types specified in step 10.

iv Select one or more NEs in the list and click on the OK button. The Selected Network Elements form closes and the NEs are listed on the Target Configuration form.

Note — If you choose an NE that is incompatible with the type of script, the 5620 SAM displays an error message and does not commit the changes when you try to close the Target Configuration form.

v Change the displayed name of a target NE, if required, by double-clicking on the Target field and entering the desired name.

vi If you created parameter tags and default values in step 8, apply the default value of each parameter to NEs, as required.

- Choose one or more NEs from the Target List panel.
- Choose the parameter from the Settings panel.
- Change the default value, if required, by double-clicking on the value and entering the default value that you want to apply to the specified targets.
- Click on the Apply To Selected button.

See Procedure 4-4 for more information about using parameter tags in CLI command scripts when you use the script manager editor.

vii If the script includes a Velocity UI header, configure the Velocity parameters in the Settings panel, as required.

viii Click on the OK button. The Target Configuration form closes and the Script (Edit) form reappears with the Targets tab displayed.

12 Go to step 14.

13 Add an instance of a XML API script.

i Click on the Instances tab button.

ii Click on the Create button. The Instance Configuration form opens.

iii Click on the Add button in the Instance List to add an XML API script instance to the list.

iv Change the name of the XML API script instance by double-clicking on the name in the Instance column of the Instance List and entering a name, if required.

v Change the description of the XML API script instance by double-clicking on the text in the Description column of the Instance List and entering new text, if required.

vi If the script includes a Velocity UI header, configure the Velocity parameters, as required.

vii Click on the OK button. The Instance Configuration form closes and the XML API Script (Edit) form reappears with the Instances tab displayed.

14 Click on the OK button. The Script (Create) form closes.

15 Close the Script Manager.

Procedure 4-12 To preview a script

Perform this procedure to view a read-only version of a script.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

- **a** To preview a CLI script, choose CLI Script (Scripting) from the object drop-down menu.
- **b** To preview an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

5 Perform one of the following:

- **a** To preview a CLI script, click on the Targets tab button.
- **b** To preview an XML API script, click on the Instances tab button.

6 Choose an entry from the list and click on the Preview button. The Preview form opens and displays a read-only version of the script that is associated with the selected target or instance. Where applicable, the parameters that were inserted in the script are displayed with the configured default values.

7 Perform the following steps to save the script, if required.

- **i** Save the script to a local or network text file by choosing File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network.
- **ii** Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.
- **iii** Click on the Export button. The script version text file is saved in the specified location. The Script (Edit) form reappears with the Targets tab displayed.

8 Choose File→Close from the Preview form menu. The Script (Edit) form reappears with the Targets tab or Instances tab displayed.

9 Click on the OK button to close the script manager.

Procedure 4-13 To preview a script with runtime parameters

Perform this procedure to view a read-only version of a script that includes runtime parameters.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

   a To preview a CLI script with runtime parameters, choose CLI Script (Scripting) from the object drop-down menu.

   b To preview an XML API script with runtime parameters, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

5 Perform one of the following:

   a To preview a CLI script with runtime parameters, click on the Targets tab button.

   b To preview an XML API script with runtime parameters, click on the Instances tab button.

6 Choose an entry from the list and click on the Preview button. The Runtime Parameter Configuration form opens and displays the runtime parameters of the script that is associated with the selected target or instance.

7 Configure the runtime parameters, as required.

8 Click on the OK button. The Preview form opens and displays a read-only version of the script that is associated with the selected target or instance. Where applicable, the parameters that were inserted in the script are displayed with the configured default values, in addition to the configured runtime parameters.

9 Perform the following steps to save the script, if required.

i Save the script to a local or network text file by choosing File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network.

ii Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.

iii Click on the Export button. The script version text file is saved in the specified location. The Script (Edit) form reappears with the Targets tab displayed.

10 Click on the OK button to close the script manager.

Procedure 4-14 To execute scripts and view and save results

Note — If a script is locked when a user creates a script instance or target, only that user can subsequently execute the instance or target.

The user name of the instance or target creator is shown in the Created By User column on the Instances or Targets tab of the script properties form.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

a To execute a CLI script, choose CLI Script (Scripting) from the object drop-down menu.

b To execute an XML API script, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Double-click on an entry in the list of scripts, or select an entry and click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed.

5 Perform one of the following to execute scripts associated with all targets or instances simultaneously, or specify the targets or instances on which to execute the associated scripts:

a To execute the scripts associated with all targets or instances simultaneously, click on the Execute button on the General tab of the *Script_type* Script (Edit) form.

- If the script has runtime parameters, the Runtime Parameter Configuration form opens. Configure the runtime parameters and click on the OK button to save the configuration and close the Runtime Parameter Configuration form.
- Go to step 6.

b To execute the scripts on specified targets or instances, click on the Targets or Instances tab button.

- Choose an entry or entries from the list.
- Click on the Execute button. If the script has runtime parameters, the Runtime Parameter Configuration form opens. Configure the runtime parameters and click on the OK button to save the configuration and close the Runtime Parameter Configuration form.
- Go to step 7.

6 Click on the Targets or Instances tab button.

7 Perform one of the following to view the results of the scripts:

a Choose a target or instance from the list. The results of the script that was executed on the specified target or instance appears in the panel below the list.

- Save the results by clicking on the Save Result button. A dialog box appears. Click on the OK button.

Note — You can save the results of multiple scripts simultaneously by selecting the entries in the list and clicking on the Save Result button.

- Export the results to a local or network text file.
- Click on the Export button. A dialog box appears and prompts you to choose a file storage location on the network.
- Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field.
- Click on the Export button. The results text file is saved in the specified location.

b Choose a target or instance from the list. The results of the script that was executed on the specified target or instance appears in the panel below the list.

- Save the results by clicking on the Save Result button. A dialog box appears. Click on the OK button.

Note — You can save the results of multiple scripts simultaneously by selecting the entries in the list and clicking on the Save Result button.

- Click on the Show Results button. The Result window opens.
- Click on the Search button. A list of scripts appears.
- Choose one or more entries from the list and click on the Properties button. The View Result window opens.

c Choose an entry or entries from the list and click on the View Selected button. The View Selected form opens and displays the results of the scripts. The results for each target are separated by a comment that indicates the script version, associated target, script status, time and date of execution, and script parameters.

- If required, export the results of the script to a local or network text file by choosing File→Export from the Editor menu, or click on the Export icon. A dialog box appears and prompts you to choose a file storage location in the network. Scroll to the location in which you want to save the text file, and enter a filename in the appropriate field. Click on the Export button. The results text file is saved in the specified location.
- Close the View Selected form by choosing File→Close from the View Selected menu.

8 Click on the OK button to close the *Script_type* Script (Edit) form.

Procedure 4-15 To view and compare script results

Perform this procedure to perform side-by-side comparisons of two different script results.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

2 Perform one of the following:

a To compare CLI script results, choose CLI Script (Scripting) from the object drop-down menu.

b To compare XML API script results, choose XML API Script (Scripting) from the object drop-down menu.

3 Configure the list filter criteria. A list of scripts appears at the bottom of the Script Manager form.

4 Choose one or more scripts from the list and perform one of the following:

a Click on the Show Results button. The Multiple Instances window opens. Go to step 7.

b Click on the Properties button. The *Script_type* Script (Edit) form opens with the General tab displayed. Go to step 5.

5 Click on the Targets tab or Instances tab button.

6 If not already done, execute the scripts, as described in Procedure 4-14. Go to step 8.

7 Configure the list filter criteria. A list of results appears at the bottom of the form.

8 Choose two results from the list to be compared and click on the Compare button. The Compare window opens with the two sets of results displayed side-by-side. The differences in the results are highlighted.

9 Scroll through the differences between the two scripts by choosing View→Next or View→Previous from the Editor menu, or by clicking on the Next or Previous icon.

10 Choose File→Close to close the Compare window.

11 Click on the OK button to close the script manager.

Procedure 4-16 To create or modify a script bundle

Perform this procedure to create a script bundle, or to modify the parameters and member list of an existing bundle.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Script Bundle (Scripting) from the drop-down menu.

3 Perform one of the following:

a To create a script bundle, click on the Create button. The Script Bundle (Create) form opens.

b Modify a script bundle.

i Click on the Search button. A list of script bundles is displayed.

ii Select a script bundle in the list and click on the Properties button. The Script Bundle (Edit) form opens.

4 Configure the parameters:

- Auto-Assign ID
- Bundle ID
- Name
- Description
- State
- Mode

Note — The Auto-Assign ID and Bundle ID parameters are configurable only during script bundle creation.

5 If you are creating a new bundle, click on the Apply button. The form refreshes, and the form name changes to Script Bundle (Edit).

6 Perform one of the following to add a script or template as a bundle member.

a Add an existing script or template to the bundle. Perform the following steps.

i Click on the Add Members button and choose a member type from the drop-down menu. The Find Script form opens and displays a list that is filtered by member type.

ii Select an entry in the list and click on the OK button. The Find Script form closes, and the chosen member is listed on the Script Bundle (Create) form.

b Create a new script or template for the bundle. Perform the following steps.

i Click on the Create button and choose a member type from the drop-down menu. A script or template creation form opens.

ii Perform the appropriate script or template creation procedure in this guide, beginning at the first step after the opening of the creation form; you must end with the step that closes the creation form.

7 Repeat step 6 to add or create a member, as required.

8 Close the Script Bundle (Edit) form.

9 Close the Script Manager form.

Procedure 4-17 To import a script bundle

Perform this procedure to import a script bundle. See "Bundle import and export" in this chapter for information about the conditions that apply to a bundle import operation.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Script Bundle (Scripting) from the drop-down menu.

3 Click on the Import button. The Open form opens.

4 Use the form to navigate to the bundle file that you want to import.

5 Select the file and click on the Open button. The Open form closes and the Import form opens to display a message about the pending import operation.

6 Click on the Continue button. The details of the import operation are displayed on the Import form as the operation progresses.

7 When the import operation is complete, as indicated on the Import form, click on the Close button. The bundle is imported to the 5620 SAM system.

8 Close the Script Manager form.

Procedure 4-18 To export a script bundle

Perform this procedure to export a script bundle to a file on the 5620 SAM client file system. See "Bundle import and export" in this chapter for information about the conditions that apply to a bundle export operation.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Script Bundle (Scripting) from the drop-down menu.

3 Click on the Search button. A list of script bundles is displayed.

4 Select a script bundle and click on the Export button. The Open form opens.

5 Use the form to navigate to the directory in which you want to save the exported bundle file.

6 Select the file and click on the Save button. The 5620 SAM saves the script bundle in a zip file at the specified location, the Open form closes, and the Export form opens to display a status message about the export operation.

7 Close the Export form.

8 Close the Script Manager form.

Procedure 4-19 To view the import and export history of a script bundle

Perform this procedure to view the history of import and export operations for a specific script bundle.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Script Bundle (Scripting) from the drop-down menu.

3 Click on the Search button. A list of script bundles is displayed.

4 Select a script bundle and click on the Properties button. The script bundle properties form opens.

5 Expand the Import/Export History panel at the bottom of the form. This panel lists each import operation for the script bundle.

6 To view an import or export entry, select the entry and click on the Properties button. The Bundle Event form opens and displays information about the operation, such as the user name associated with the operation, and the operation type, time, and status.

7 Close the Bundle Event form, if it is open.

8 Close the script bundle properties form.

9 Close the Script Manager form.

Procedure 4-20 To delete a script bundle

Perform this procedure to delete a script bundle.

Note — When you delete a script bundle, the member scripts are also deleted.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

3 Choose Script Bundle (Scripting) from the drop-down menu.

4 Click on the Search button. A list of script bundles is displayed.

5 Select a script bundle and click on the Delete button. A dialog box appears.

6 Select the check box to confirm that you understand the implications of the action.

7 Click on the Yes button. The 5620 SAM deletes the script bundle.

8 Close the Script Manager form.

Procedure 4-21 To manage the CLI scripts of a specific NE

1 From the navigation tree, select Equipment from the view selector.

2 Locate the device on which you want to manage scripts.

3 Right-click on the device icon and choose Properties from the contextual menu. The Network Element (Edit) form opens with the General tab displayed.

4 Click on the Scripts tab button. A list of all the CLI scripts that are associated with the NE is displayed.

5 Perform one or more of the following:

a Perform the following to execute a script or scripts:

i Choose the script or scripts that you want to execute from the list.

ii Click on the Execute button. If the script includes runtime parameters, the Runtime Parameter Configuration form opens. Configure the runtime parameters in the Settings panel and click on the OK button to save the configuration and close the form.

The results of the execution are displayed in the panel below the list.

b Perform the following to view or configure the general parameters, versions, and target configuration of a script:

i Choose a script from the list.

ii Click on the Script button. The Script (Edit) form opens with the General tab displayed.

iii Configure the parameters:

- Name
- Description
- Type
- Use Latest Version
- Starting Point
- Continue On Command Failure
- Mode

iv Click on the Versions tab button.

v Perform steps 7 to 10 of Procedure 4-3 to add a new version of the script.

vi Click on the Targets tab button.

vii Perform steps 6 to 13 of Procedure 4-9 to add or remove targets from the script.

viii Click on the OK button to close the Script (Edit) form.

c Manage the results of scripts by clicking on the Results button. See Procedure 4-15 for more information about how to view and compare the results of scripts.

d Open the script manager by clicking on the Script Manager button. Procedures 4-1 to 4-9 and 4-11 to 4-15 describe the actions you can perform from the script manager.

Procedure 4-22 To manage the CLI scripts of a specific service site

Perform this procedure to manage the CLI scripts of a service site from the properties form of the site.

1 Choose Manage→Services from the 5620 SAM main menu. The Manage Services form opens.

2 Choose Site (Service Management) from the object drop-down menu and specify a filter for the search, if required.

3 Click on the Search button. A list of service sites appears.

4 Select a site and click on the Properties button. The Site (Edit) form opens with the General tab displayed.

5 Click on the Scripts tab button. A list of all the CLI scripts that are associated with the site is displayed.

6 Perform one or more of the following steps.

a Perform the following steps to execute a script or scripts by performing the following steps.

i Choose the script or scripts that you want to execute from the list.

ii Click on the Execute button. If the script includes runtime parameters, the Runtime Parameter Configuration form opens. Configure the runtime parameters in the Settings panel and click on the OK button to save the configuration and close the form.

The results of the execution are displayed in the panel below the list.

b Perform the following steps to view or configure the general parameters, versions, and target configuration of a script.

i Choose a script from the list.

ii Click on the Script button. The Script (Edit) form opens with the General tab displayed.

iii Configure the parameters:

- Name
- Description
- Type
- Use Latest Version
- Starting Point
- Continue On Command Failure
- Mode

iv Click on the Versions tab button.

v Perform steps 7 to 10 of Procedure 4-3 to add a new version of the script.

vi Click on the Targets tab button.

vii Perform steps 6 to 13 of Procedure 4-9 to add or remove targets from the script.

viii Click on the OK button to close the Script (Edit) form.

c Manage the results of scripts by clicking on the Results button. See Procedure 4-15 for more information about how to view and compare the results of scripts.

d Open the script manager by clicking on the Script Manager button. Procedures 4-1 to 4-9 and 4-11 to 4-15 describe the actions you can perform from the script manager.

Procedure 4-23 To configure a script scheduled task

Perform this procedure to assign a 5620 SAM schedule to a script task. This procedure assumes the following:

- You have created at least one version of the script.
- You have added at least one target or instance for the script.
- You have created a 5620 SAM schedule. See the *5620 SAM User Guide* for more information about 5620 SAM schedules.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Perform one of the following:

a Assign a schedule to a CLI script task.

i Choose CLI Script (Scripting) from the object drop-down menu.

ii Configure the filter criteria, if required, and click on the Search button.

iii Select a script from the list and click on the Properties button. The CLI Script (Edit) form opens with the General tab displayed.

iv Click on the Targets tab button.

v Configure the filter criteria, if required, and click on the Search button.

vi Select a target from the list and click on the Schedule button. The Script Scheduled Task (Create) form opens with the General tab displayed.

b Assign a schedule to an XML API script task.

i Choose XMLAPI Script (Scripting) from the object drop-down menu.

ii Configure the filter criteria, if required, and click on the Search button.

iii Select a script from the list and click on the Properties button. The XLMAPI Script (Edit) form opens with the General tab displayed.

iv Click on the Instances tab button.

v Configure the filter criteria, if required, and click on the Search button.

vi Select an instance from the list and click on the Schedule button. The Script Scheduled Task (Create) form opens with the General tab displayed.

3 Configure the parameters:

- Scheduled Task Name
- Scheduled Task Description
- Save Results
- Administrative State

4 Click on the Select button in the Schedule panel. The Select Schedule - Script Scheduled Task form opens.

5 Configure the filter criteria, if required, and click on the Search button.

6 Select a schedule from the list and click on the OK button. The Select Schedule - Script Scheduled Task from closes.

7 Click on the OK button. The Script Scheduled Task (Create) form closes.

If this is the first script scheduled task created for this script, the *Script_type* Script (Edit) form refreshes with the Schedule tab.

8 Close the *Script_type* Script (Edit) form.

Procedure 4-24 To monitor a script scheduled task

Perform this procedure to monitor a script scheduled task. This procedure assumes that you have assigned a schedule to a script task.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Perform one of the following:

   a Monitor a CLI script scheduled task.

      i Choose CLI Script (Scripting) from the object drop-down menu.

      ii Configure the filter criteria, if required, and click on the Search button.

      iii Select a script from the list and click on the Properties button. The CLI Script (Edit) form opens with the General tab displayed.

   b Monitor an XML API script scheduled task.

      i Choose XMLAPI Script (Scripting) from the object drop-down menu.

      ii Configure the filter criteria, if required, and click on the Search button.

      iii Select a script from the list and click on the Properties button. The XLMAPI Script (Edit) form opens with the General tab displayed.

3 Click on the Schedule tab button.

4 Configure the filter criteria, if required, and click on the Search button.

5 Select a script scheduled task from the list and click on the Properties button. The Script Scheduled Task (Edit) form opens with the General tab displayed.

6 View the Execution Status. The value is one of the following:

- None
- In Progress
- Skip Requested
- Skipped
- Stop Requested
- Stopped
- Succeeded
- Failed
- Delay Requested
- Delayed
- Started

7 View the Status. The value is one of the following:

- Running—The scheduled task is running.
- Not Running—The scheduled task is not running.
- Completed—The scheduled task is finished running.

8 Close the Script Scheduled Task (Edit) form.

9 View the script results, as required. See Procedure 4-15 for more information.

10 Close the *Script_type* Script (Edit) form.

11 Close the Script Manager form.

5 — Script cascading

5.1 Script cascading overview

The 5620 SAM script cascading function allows you to create a custom workflow for a task that requires the controlled execution of multiple scripts. For example, you can configure scripts to invoke other scripts and templates, pass variables to the scripts, and determine whether to invoke subsequent scripts based on the script results. The passing of control between scripts to perform a task is called an execution flow.

To simplify the management of multiple scripts, such as in script cascading, you can group the scripts and templates in a script bundle. Script cascading does not require a script bundle, but using a bundle facilitates the import and export of multiple scripts, and exercises version control. See chapter 4 for information about creating and using script bundles.

Caution — A script that is not correctly created or applied can seriously damage the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage and ensure that each script is verified and validated before the script is run on a device in a live network.

5.2 Control scripts

Script cascading requires a type of script called a control script. A control script invokes, or calls, other scripts and templates, passes variables to the scripts and templates, and determines the subsequent flow based on the results.

A control script has a one-to-many association with other scripts and templates through bindings to specific script and template versions. A 5620 SAM operator creates a binding during script configuration. By default, a control script executes the latest versions of the scripts and templates to which the control script is bound.

Control scripts, which are written in the Velocity language, can include conditions such as the following:

- actions to perform, based on previous script execution results
- which variables to pass to other scripts
- how to define and format returned values from other scripts

Note — Sample control scripts are available. See chapter 4 for information about how to browse the sample scripts.

5.3 Script bindings

An execution flow can contain multiple control scripts. The binding that you create between scripts determines whether a control script is an invoker control script or a handler control script. An invoker control script calls other scripts. A CLI script or an XML API script can pass control to a handler control script, which is optional. A handler control script determines the flow based on the output of the calling script, and facilitates the execution of subsequent scripts as required.

An invoker control script initially opens an operator-created GUI to obtain parameter values that the script uses to set variable values. A control script can export variables so that they are available to all other invoked control scripts, or pass variables to specific control, CLI, or XML API scripts, as required.

You can designate an invoker control script as a starting-point script, which is a script that initiates an execution flow. An execution flow can contain multiple starting-point scripts. For example, one script initiates an entire service-creation execution flow that involves creating sites and SAPs; another script initiates only the portion of the flow for SAP creation.

5.4 Developing control scripts

An invoker control script contains a Velocity UI header and Velocity logic that uses GUI input to set variables. A control script uses the standard Velocity properties and the alCtrl variable, which is only for cascading functions. See section 3.6 for more information about creating Velocity UI headers and using Velocity properties. See section 3.7 for information about using the VTL, and for a complete list of the methods available to 5620 SAM control scripts.

Before a control script can invoke another script, an invoker for the script must be defined by using one of the following alCtrl methods:

- for a CLI script invoker:
  #set($myInvoker = $alCtrl.defineCliScriptInvoker('))
- for a control script invoker:
  #set($myInvoker = $alCtrl.defineControlScriptInvoker('))
- for an XML API script invoker or a template invoker:
  #set($myInvoker = $alCtrl.defineXmlApiScriptInvoker('))

See section 3.7 for a list of all alCtrl methods and a description of each method.

To invoke a script after the invoker is defined, use the following method:

```
$myInvoker.invokeScript()
```

You can define the invoker in a local variable, such as $myInvoker, which is used in the following and subsequent examples:

```
#set( $myInvoker = $alCtrl.defineCliScriptInvoker('CLI Script Name')
)

#set( $myInvoker = $alCtrl.defineControlScriptInvoker('Control Script
Name') )

#set( $myInvoker = $alCtrl.defineXmlApiScriptInvoker('XML API Script
Name') )
```

After the variable is defined, an XML API script or a control script is invoked using the following method, in which the script name is implicitly passed as a parameter:

```
$myInvoker.invokeScript()
```

To run the CLI script invoker, you must specify the target NE ID as a parameter, as shown in the following example:

```
$myInvoker.invokeScript('NE FDN or IP address)
```

Calling a script does not open the Velocity GUI defined in the script unless the following method is used:

```
$alCtrl.exitNow($myInvoker)
```

The method ends the execution of the current script, passes control to the calling script, and displays the Velocity GUI that is defined in the script.

5.5 Passing variables between scripts

Managing script invoker variables requires 5620 SAM ScriptInvoker methods, which support the following variable types:

- bitmask
- boolean
- combobox
- date
- float
- format
- integer
- largeString
- long
- multiSelection
- password
- range
- selection
- string

A control script can use ScriptInvoker methods to pass variables directly to specified scripts. The syntax for passing a variable differs, depending on the scenario, as shown in the following examples.

To pass a literal value:

```
$myInvoker.addVariable('type', 'name', 'value')
```

To pass a current variable value:

```
$myInvoker.addVariable('type', 'name')
```

To pass a previously defined variable:

```
$myInvoker.addVariable($otherVariable)
```

Note — If a variable is not passed to a script but is in the script, the variable is assigned the default value, which is specified in the Velocity header of the invoked script.

A control script can also export variables so that the variables are available to each control script in the execution flow. Exported variables are not available to CLI or XML API scripts. The following examples show the syntaxes for passing variables in different scenarios.

To export a literal value:

```
$myInvoker.exportVariable('type', 'name', 'value')
```

To export a current variable value:

```
$myInvoker.exportVariable('type', 'name')
```

To export a previously defined variable:

```
$myInvoker.exportVariable($otherVariable)
```

See section 3.7 for a list of all ScriptInvoker methods and a description of each method.

5.6 Cascade execution

The 5620 SAM GUI displays an execution flow as the flow progresses. Each invoked script and script result appear in a tree view. You can stop the execution flow, if required. Stopping a flow prevents the invocation of subsequent scripts, but does not stop the currently running scripts.

The 5620 SAM cannot automatically roll back the changes that result from executing a flow. However, you can use the cascading function to perform a rollback using scripts that you create.

The following sequence of events describes the initialization of a simple execution flow.

1 A 5620 SAM GUI operator opens a control script execution form that displays the custom Velocity GUI of the control script and an Execute button.

2 The operator uses the Velocity GUI to enter the required information.

3 The operator clicks on the Execute button.

4 The invoker control script uses the operator information to set local variables that are passed to specific scripts or exported to the other control scripts in the flow.

5 The invoker control script invokes other scripts and templates, as defined in the control script logic.

6 If another control script is invoked, it opens a GUI for additional operator input, if required, and invokes other scripts, as required.

7 When the execution of an invoked script or template is complete, the script or template performs one of the following:

- passes control to a handler control script, if there is a binding to the script
- passes control to the invoker control script

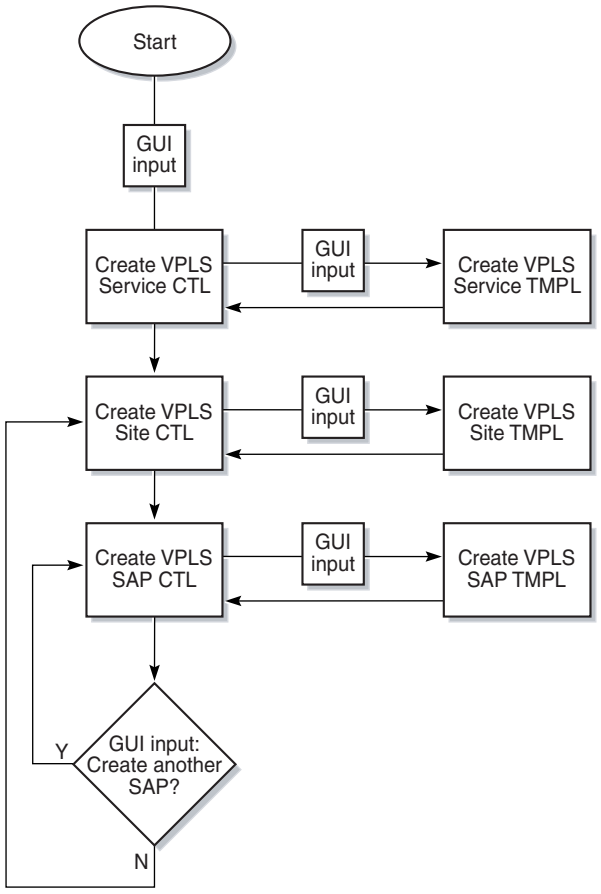
8 The handler or invoker control script performs one of the following:

- uses the execution result from the invoked script or template to prepare variables for subsequent scripts, and invokes the scripts, as required
- completes execution and exits

Note — Sample script bundles are available. Each bundle contains the scripts required to implement an execution flow for a specific task. You can customize these bundles to meet your requirements. See chapter 4 for information about how to browse the sample scripts and bundles, and Appendix B for script bundle example descriptions.

The execution flow shown in Figure 5-1 represents the flow of the VPLS Service Bundle that is listed on the Browse Examples of Scripts form, and uses the same script names.

Figure 5-1 VPLS Service Bundle execution flow



The following is the sequence of events for the example.

1 A 5620 SAM operator executes the Create VPLS Service CTL script, which opens a GUI and obtains service configuration information from the operator.

2 The Create VPLS Service CTL script invokes the Create VPLS Service TMPL template, which creates the service.

3 The Create VPLS Service CTL script passes control and the service FDN to the Create VPLS Site CTL script, which opens a GUI and obtains site and SAP configuration information from the operator.

4 The Create VPLS Site CTL script invokes the Create VPLS Site TMPL template, which creates the site.

5 The Create VPLS Site CTL script passes control and the site FDN to the SAP-creation control script, Create VPLS SAP CTL.

6 The Create VPLS SAP CTL script invokes the Create VPLS SAP TMPL template, which creates the SAP.

7 The Create VPLS SAP CTL script opens a GUI and prompts the operator to create an additional SAP on the site.

8 If the operator accepts, the flow continues from step 6.

9 The Create VPLS SAP CTL script passes control to the Create VPLS Site CTL script.

10 The Create VPLS Site CTL script prompts the user to create an additional site.

11 If the operator accepts, the flow resumes at step 4.

12 The VPLS Site CTL script passes control to the Create VPLS Service CTL script.

13 The Create VPLS Service CTL compiles the results.

14 The Create VPLS Service CTL exits.

5.7 Managing execution results

A control script can manage the following types of execution results:

- result items that are tabulated in the script execution window
- result objects that contain values that are passed to the invoking control script
- text that is displayed in the script execution window

Preparing result items and objects

When the execution of an invoked script is complete, the 5620 SAM creates a Java object that contains the execution result. You can use the alCtrl variable and the following methods to handle execution results:

- addResultItem—adds a result entry to the table in the script execution window
- addReturnValue—passes a result value to the invoking control script

See section 3.7 for a list of all alCtrl methods and a description of each method.

addResultItem method

The Script Results table entry that the addResultItem method creates contains a text description and an optional object FDN. When an FDN is included as the method parameter, you can directly open the properties form of the object from the script execution window.

The addResultItem method syntax is the following:

```
$alCtrl.addResultItem('Text description')
```

or

```
$alCtrl.addResultItem('Text description', 'object_FDN')
```

addReturnValue method

When you use the addReturnValue method to define a result value, the invoking script can use the value to determine whether the execution is successful or failed, and then perform the appropriate action. For example, if the invoked script successfully creates a service, the invoking script can make the returned service FDN available to other scripts that create the service SAPs and SDPs.

The addReturnValue method syntax is the following:

```
$alCtrl.addReturnValue('variable_name', $local_variable)
```

or

```
$alCtrl.addReturnValue('variable_name', 'value_text')
```

Processing script results

You can use the 5620 SAM ScriptResult methods, which are listed and described in section 3.7, to perform result management functions such as the following.

- Return one or all items from a result object.
- Return the result file that contains the result objects.
- Return the execution success or failure status.

The following example shows the syntax for defining a variable to store a script result and then calling a ScriptResult method to return the error messages in the result.

```
#set( $myResult = $myInvoker.invokeXmlApiScript() )

$myResult.errorMessage
```

Displaying result text in the 5620 SAM GUI

The content of the text panel in the script execution window displays, in plain text or HTML format, the execution result text specified in a control script. The text display is enabled when the first non-empty line in a control script is one of the following:

- Content-Type: text/html
- Content-Type: text/plain

If a Content-Type value is not specified, or is specified on a line other than the first non-empty line, the value defaults to text/plain. An OSS application can specify any format that is required, such as XML or comma-separated.

Code 5-1 shows an example that displays result text in HTML format; the Content-Type value for the example is text/html.

Code 5-1: HTML result display example

```
<B>Boldface text</B>
<P>Normally formatted paragraph text</P>
<a href="hyperlink">hyperlink text</a>
```

The *hyperlink* value in Code 5-1 is a link to an object properties form, and contains the object FDN. For example, network:192.168.3.4 specifies the NE at IP address 192.168.3.4. When a GUI operator clicks on the hyperlink, the properties form for the NE opens. The *hyperlink text* value is a descriptor that is displayed beside the hyperlink, such as "properties form".

Reading a CLI script result file

CLI script execution generates a ResultFile object that contains a result file. The getResultFile method returns the result file from an ResultFile object. You can then use an expression like the following to return an iterator for the result file:

```
#set($lineIterator = $resultFile.lineIterator('reg_exp') )
```

This expression returns an instance of ResultLineIterator, which iterates each line that matches *reg_exp*, a regular-expression string. If *reg_exp* is not specified, ResultLineIterator iterates each line of the file. You close an ResultLineIterator instance using the close() method.

An ResultFile object also supports the following methods:

$resultFile.getSize()—returns the size of the result file, which can affect the 5620 SAM processing load

$resultFile.close()—closes all open iterators of a file

Note — Alcatel-Lucent recommends that you close each iterator instance individually using the close() method. However, you can call the $resultFile.close() method at any point to ensure that all iterators of a file are closed, if required.

See section 3.7 for a list of all ResultFile methods and a description of each method.

Result-line iterator usage

You can use a result-line iterator only once. If you do not close an iterator after it finishes, iterator remains open until the control script execution completes.

To loop through the lines that an iterator returns, a control script must use the Velocity iterator tool called $vmIterator. The example in Code 5-2 uses this tool to return each line of the file, one line at a time:

Code 5-2: Simple iterator looping example

```
#set( $lineIterator = $resultFile.lineIterator() )
#set( $lines = $vmIterator.wrap( $lineIterator ) )
#foreach( $line in $lines )
$lines.more()
#end
$lineIterator.close()
```

To exit a loop before the last result line is read, you can insert logical statements after the #foreach statement to determine whether additional lines require processing, and call $lines.more() only if this condition is true.

5.8 Debugging

You can configure the 5620 SAM to record the local and exported variable values from a flow to the 5620 SAM server log before and after the execution of each script. The values are in Debug-level log entries. To enable the logging function, you must enable debug-level logging for the following Java class:

com.timetra.nms.server.script.ScriptExecutionLogger

5.9 Workflow for script cascading

The following workflow describes the high-level steps that are required to manage script cascading after you do the following.

- Create a workflow to define the required tasks in the execution flow.
- Create, modify, or import CLI scripts, XML API scripts, or templates for the tasks, as required.

1 Create one or more control scripts to prompt for operator input and control the execution flow; designate scripts as starting-point scripts, as required.

2 Create invoker and handler bindings between scripts, as required.

3 Initiate the execution flow.

4 Preview the execution flow to ensure that it is correct.

5 When the execution flow is correct, switch the mode of each script and template to Released.

6 Start the execution flow and monitor the progress.

7 Review, and save or export the execution results.

5.10 Script cascading procedures

Use the following procedures to perform script cascading functions.

Procedure 5-1 To create a control script

Perform this procedure to create a control script for an execution flow.

Note — See chapter 4 for information about creating CLI scripts and XML API scripts. See chapter 6 for information about creating templates.

Warning — Scripts that are not correctly applied or created can seriously damage the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage, and ensure that each script is verified and validated before it is executed on a device in a live network.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose Control Script (scripting) from the object drop-down menu.

3 Click on the Create button. The Control Script (Create) form opens with the General tab displayed.

4 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- Starting Point
- Mode

5 Perform the following steps to add the script to a script bundle, if required.

i Click on the Select button in the Member of Bundle panel. The Select Member of Bundle form opens.

ii Click on the Search button. A list of script bundles is displayed.

iii Select a script bundle in the list and click on the OK button. The bundle name and ID are displayed on the Control Script (Create) form.

6 Click on the Apply button. The form refreshes with additional tabs, and the form name changes to Control Script - *script_name* (Edit).

7 Click on the Versions tab button.

8 Click on the Create button. The Script Editor form opens.

9 Perform one of the following.

a Import an existing text file that contains the script text. Perform the following steps.

i Choose File→Import from the Editor menu. The Import form opens.

ii Use the form to navigate to the required file and click on the Import button. The script appears in the script manager editor workspace.

iii Modify the script as required.

Note — Sample scripts are available. See Procedure 4-7 for information about how to browse and create sample scripts, and Appendix B for script examples.

b Enter the script text in the script manager editor workspace.

10 Perform one of the following to save the version of the CLI or XML API script:

a Save the script to the Script Manager.

i Choose File→Save from the Editor menu or click on the Save icon. The Comment dialog box appears.

ii Configure the Bundle ID parameter.

iii Click on the OK button.

b Export the script to a text file.

i Choose File→Export from the Editor menu, or click on the Export icon. The Export form opens.

ii Navigate to the location in which you want to save the file, and enter a filename in the File Name field.

iii Click on the Export button. The script text is saved in the specified file.

11 Choose File→Close from the Editor menu. The Script Editor form closes and the Control Script - *script_name* (Edit) form lists the new script version.

12 Perform the following steps to create a script binding.

i Click on the Executables tab button.

ii Right-click on the appropriate script object and choose Bind Executable Script. The Select Script(s) to Create Binding form opens.

Note — To create a binding between the control script and another script or template, you must right-click on the Control Script object. To create a binding between a child script of the control script and another script or template, you must right-click on the child script object.

iii Select a script in the list and click on the OK button. A child script object is displayed below the script object to which it is bound.

13 Repeat step 12 to bind another script to the control script, if required.

14 Perform the following steps to view the spans of control that are applied to the script, if required.

i Click on the Spans tab button.

ii Choose a span from the list.

iii Click on the Properties button. The Span – *span_name* (Edit) form opens.

iv Click on the following tab buttons, as required:

- Contents, to view all scripts to which the span of control is applied
- Span of Control Profiles, to view the span of control profiles that are applied to the span of control
- User Groups, to view the user groups with which the span of control profiles are associated
- Users, to view the users that belong to the user groups

v Close the Span – *span_name* (Edit) form.

15 Close the Control Script - *script_name* (Edit) form.

Procedure 5-2 To create a script binding

Perform this procedure to create an invoker or handler binding between a control script and a CLI script, XML API script, another control script, or template.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose one of the following from the object drop-down menu:

- CLI Script (scripting)
- Control Script (scripting)
- Template (scripting)
- XML API Script (scripting)

3 Click on the Search button. A list of scripts or templates is displayed.

4 Select a script or template in the list and click on the Properties button. The properties form for the script or template opens with the General tab displayed.

5 Click on the Executables tab button.

6 To bind a handler control script to a CLI script, XML API script, or template, right-click on the script or template object and choose Bind Control Script. The Select Script(s) to Create Binding form opens.

7 To bind a control script to a control script, right-click on the Control Script object and choose Bind Executable Script. The Select Script(s) to Create Binding form opens.

8 Select a script in the list and click on the OK button. A child script object is displayed below the script object to which it is bound.

Procedure 5-3 To start an execution flow

Perform this procedure to initiate the execution of the scripts in an execution flow.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Perform one of the following.

a Start an execution flow from a bundle. Perform the following steps.

i Choose Script Bundle (scripting) from the drop-down menu.

ii Click on the Search button. A list of script bundles is displayed.

iii Select a script bundle in the list and click on the OK button. The Script Bundle (Edit) form opens and displays a list of the scripts and templates in the bundle.

iv Select a control script that has the Starting Point indicator selected and click on the Execute Script button. The Execute Script: '*script_name*' form opens. The left panel displays the execution flow in a tree view. If the control script is configured to open a GUI form, the GUI form is displayed in the right panel.

Note 1 — You can click on the Preview Script button at any tine to view the current script progress, if required.

Note 2 — You can click on the Stop button at any tine to stop the flow execution, which stops when the currently running script finishes execution.

b Start an execution flow from a control script that is not in a bundle. Perform the following steps.

i Choose Control Script (scripting) from the drop-down menu.

ii Click on the Search button. A list of control scripts is displayed.

iii Select a control script in the list and click on the Initiate Script Flow button. The Execute Script: '*script_name*' form opens.

3 If a GUI form is displayed, enter the required information on the form, and then click on the Execute button to resume the execution flow.

4 If a script or template object in the left pane has a red icon, the object has encountered a problem. Perform the following steps.

   i Click on the object to display the Status and Script Results panels for the flow.

   A flow has a Status of Failed if no script or template execution is successful, and a Status of Partially Failed if the execution of only some scripts or templates is successful.

   ii View the result information in the Script Results panel and determine the corrective action to take during the next execution attempt.

   iii To restart the execution, click on the object above the object with the red icon and perform the required corrective action, such as correcting a parameter value on a GUI form.

5 When the execution of the flow is complete, perform the following steps to save a script result object to the 5620 SAM database, if required.

Note — Each script execution generates a new script result object.

   i Select an object in the left pane.

   ii Click on the Save Results button. A dialog box appears.

   iii Click on the OK button. The selected execution result object is saved.

6 When the execution of the flow is complete, perform the following steps to view the saved script results, if required.

   i Click on the Show Results button. The Results form opens.

   ii Click on the Search button. A list of result objects is displayed.

   iii Select a result object and click on the Properties button. The View Result form opens and displays the result text.

   iv View the result.

   v Close the View Result form.

7 When the execution of the flow is complete, perform the following steps to export a saved script result object to a text file, if required.

   i Click on the Show Results button. The Results form opens.

   ii Click on the Search button. A list of result objects is displayed.

   iii Select a result object and click on the Export button. The Export form opens.

   iv Use the form to specify the location, name, and type of the file that is to contain the result object.

v Click on the Export button. The Export form closes, and the file is saved to the specified location.

vi Close the Results form.

8 When the execution of the flow is complete, perform the following steps to compare two result objects, if required.

i Click on the Show Results button. The Results form opens.

ii Click on the Search button. A list of result objects is displayed.

iii Select two result objects and click on the Compare button. The Compare form opens and displays the result objects in different panels. An orange triangle icon marks each line that differs between the objects.

iv View the comparison result.

v Close the Compare form.

9 When the execution of the flow is complete, perform the following steps to delete one or more result objects, if required.

i Click on the Show Results button. The Results form opens.

ii Click on the Search button. A list of result objects is displayed.

iii Select one or more result objects and click on the Delete button. A dialog box appears.

iv Click on the Yes button. The result objects are deleted.

10 Close the open forms, as required.

Procedure 5-4 To manage script results

Perform this procedure to do the following for a previously run script or execution flow:

- View script results.
- Export script results to a file.
- Compare script results.
- Delete script results.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose the appropriate script or template object type from the drop-down menu.

3 Click on the Search button. A list of scripts or templates is displayed.

4 Select an entry in the list and click on the Show Results button. A list form opens.

5 Click on the Search button. A list of result objects is displayed.

6 Perform the following steps to view the script results, if required.

- i Select a result object and click on the Properties button. The View Result form opens and displays the result text.
- ii View the result.
- iii Close the View Result form.

7 Perform the following steps to export a script result object to a text file, if required.

- i Select a result object and click on the Export button. The Export form opens.
- ii Use the form to specify the location, name, and type of the file that is to contain the result object.
- iii Click on the Export button. The Export form closes, and the file is saved to the specified location.
- iv Close the Results form.

8 Perform the following steps to compare two result objects, if required.

- i Select two result objects and click on the Compare button. The Compare form opens and displays the result objects in different panels. An orange triangle icon marks each line that differs between the objects.
- ii View the comparison result.
- iii Close the Compare form.

9 Perform the following steps to delete one or more result objects, if required.

- i Select one or more result objects and click on the Delete button. A dialog box appears.
- ii Click on the Yes button. The result objects are deleted.

10 Close the list form.

11 Close the Script Manager form.

Procedure 5-5 To delete a script binding

Perform this procedure to remove a binding between two scripts in an execution flow.

1 Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager form opens.

2 Choose the appropriate script or template object type from the drop-down menu.

3 Click on the Search button. A list of scripts or templates is displayed.

4 Select an entry in the list and click on the Properties button. The script properties form opens with the General tab displayed.

5 Click on the Executables tab button.

6 Right-click on the script that has the binding to delete, and choose Remove Script Binding from the contextual menu. A dialog box appears.

7 Click on the Yes button. The binding is removed; if there are no more bindings between the object and another object, the object is also removed from the tree view on the Executables tab.

8 Close the object properties form.

9 Close the Script Manager form.

6 — XML API template administration

6.1 XML API template overview

The 5620 SAM supports the configuration of templates using XML API script-based templates for the following:

- services
- tunnels
- equipment

The XML API configuration templates simplify tunnel and service creation or modification by reducing the steps required to create or modify a tunnel or service component. You can use an existing tunnel or service as a starting point to create an XML API template or you can also create a template without using a managed tunnel or service object as a basis.

Service template configuration supports the configuration of VLL, VPLS, IES, VLAN, mirror, and VPRN services using XML API script-based templates.

Tunnel templates allow users to automate the creation and modification of dynamic LSPs, LSP paths, and SDPs as routers and topology changes in the network. You can create XML API configuration templates for LSP and SDP tunnels and child objects of tunnels, such as LSP paths. You can include object categories, class names, object types, and other tunnel-related objects.

Tunnel and service template configuration follows the same rules. The templates allow users to define common characteristics for a service or tunnel, or templatable service or tunnel object, and the parameter values that can be configured. Table 6-1 lists the tunnel and service templates.

Table 6-1 Service and tunnel template configuration

Service templates	Tunnel templates
VLL	LSP
VPLS	LSP Path
IES	SDP
VLAN	—
Mirror	—
VPRN	—

For tunnel and service template configuration, the 5620 SAM generates an XML API script with Velocity properties which you must modify to generate a UI for the configuration form that is tailored to your requirements and to NE-specific properties and attributes.

Equipment template functionality allows for the creation of auxiliary alarm definitions, which is specific to the 7705 SAR auxiliary alarm daughter card. You can use the template framework described in this guide to configure the auxiliary alarm daughter card. See section 16.14 for general information about auxiliary alarm daughter cards

See chapter 3 for more information about XML API scripts and Velocity.

6.2 Managing templates

You can create, modify, or delete XML API configuration templates using the Manage Templates form. The form allows you view all templates for tunnels, services, sites, and templatable child network objects. You can also use the form to convert standard service templates that were created in earlier releases to XML API templates.

A service template can use secondary, or child, templates to further define the configuration information for the created service. Templates for sites can be bound to the service template. Network objects, such as SDP bindings and interfaces, are considered child templates which can be bound to the site template. Only site templates can be bound to service templates. When you create an XML API configuration template for a service, you can choose to create templates for child objects of the service and to bind the child templates to the service template. You can bind a child template to multiple parent templates. For example, a site template can be bound to several service templates.

A configuration template can be used to create new network objects, or modify existing ones. You use the Command Type parameter to specify the type of template—creation or modification. Modification templates are secondary templates that can be bound to the creation templates. Multiple modification templates can be bound to multiple creation templates.

Warning — Alcatel-Lucent recommends that you convert old service templates to XML API configuration template only one time. If you convert a template that has already been converted, a new template is generated. However, the association of templated objects — such as services—with the original template is removed and associated with the newly converted template.

Note 1 — You must convert standard service templates that you created using Release 5.0 or earlier of the 5620 SAM to XML API service templates. All child templates that are bound to the service templates are automatically converted when you convert the service template to an XML API configuration template.

See Procedure 6-1 for information about displaying the following:

- list of sample templates
- categories that can be assigned to service templates
- template format information
- sample templates

The association of the service template with service objects that may have been created using the template in Release 5.0 or earlier of the 5620 SAM is maintained after you convert the service template. Because the 5620 SAM, Release 5.0 or earlier, does not support an association between site templates, SDP binding templates, or access interface templates with network objects that were created from the templates, no association is maintained after the conversion.

See Procedure 6-2 for more information about how to convert standard service templates to XML API configuration templates.

Note 2 — In a Template (Create) or Template (Edit) form, an asterisk on a tab button or panel title bar indicates that a field contains incorrect data, or that a mandatory field requires data.

6.3 XML API template script format

There are two main types of templates: creation templates and modification templates.

Creation template script format

Consider the following when you create an XML API configuration template script:

- the generic.GenericObject.configureChildInstance method is the only method that you can use in XML requests
- you must configure only the deployer, distinguishedName, and childConfigInfo parameters of the generic.GenericObject.configureChildInstance method. All other parameters of the are ignored, such as synchronousDeploy, deployRetries and clearOnDeployFailure, and must not be specified.
- you must set the deployer parameter to immediate
- you must set the value of the distinguishedName parameter to the fully distinguished name of the parent object

- use the getObjectFullName() method with the $parent Velocity variable to retrieve the fully distinguished name of the parent object. The Velocity variable $parent is set by the system and can be used in the script whenever the parent context is required
- you must add the value of the childConfigInfo parameter for templatable classes only, such as services, sites, or access interfaces

Code 6-1 shows a sample format of the XML API configuration template script.

Code 6-1: Sample format of the XML API configuration template

```
<xmlapiRequest xmlns="xmlapi_1.0">
<generic.GenericObject.configureChildInstance xmlns="xmlapi_1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <deployer>immediate</deployer>
    <distinguishedName>$parent.getObjectFullName()</distinguishedNam
e>
    <childConfigInfo>
      ## insert the child configuration information here
    </childConfigInfo>
  </generic.GenericObject.configureChildInstance>
</xmlapiRequest>
```

See Procedure 6-1 for information about listing the sample templates.

Modification template script format

Consider the following when you create an XML API configuration template script:

- the generic.GenericObject.configureInstance method is the only method that you can use in XML requests
- you must configure only the deployer, distinguishedName, and configInfo parameters of the generic.GenericObject.configureInstance method. All other parameters of the are ignored, such as synchronousDeploy, deployRetries and clearOnDeployFailure, and must not be specified.
- you must set the deployer parameter to immediate
- you must set the value of the distinguishedName parameter to the fully distinguished name of the configured object
- use the getObjectFullName() method with the $this Velocity variable to retrieve the fully distinguished name of the object being configured. The Velocity variable $this is set by the system and can be used in the script whenever the configured object is required in the context
- you must add the value of the childConfigInfo parameter for templatable classes only, such as services, sites, or access interfaces

Code 6-1 shows a sample format of the XML API configuration template script used to modify existing object.

Code 6-2: Sample format of the XML API configuration template used to modify existing object

```
<xmlapiRequest xmlns="xmlapi_1.0">
<generic.GenericObject.configureInstance xmlns="xmlapi_1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
    <deployer>immediate</deployer>
    <distinguishedName>$this.getObjectFullName()</distinguishedName>
    <configInfo>
      ## insert the child configuration information here
    </configInfo>
  </generic.GenericObject.configureInstance>
</xmlapiRequest>
```

6.4 Workflow to manage XML API templates

1. Convert service templates created with the 5620 SAM, Release 5.0 or earlier, to XML API service templates. You can delete old templates, if required.
2. Create XML API service or tunnel templates.
3. Create an XML API site template for service templates, if required.
4. Create XML API templates for child network objects.
5. Bind an XML API site template to an XML API service template, if required.
6. Bind children network object XML API templates to XML API site templates.
7. Apply XML API tunnel templates to Auto tunnels, if required.

6.5 XML API template management procedures

This section contains procedures about how to create templates for tunnels, services, sites, and other templatable child network objects.

Procedure 6-1 To view the sample templates and template development information

Perform this procedure to display the following:

- list of sample templates
- categories that can be assigned to service templates
- template format information
- sample templates

1. Choose Help→User documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
2. Expand the 5620 SAM Scripts and Templates Developer Guide link.
3. Expand the 5620 SAM Scripts and Templates Developer Guide link below the About this document link.

4 Expand the Template Development Information link.

5 Perform one of the following;

a To display the list of templatable objects, click on the Classes that can be created from templates link. The templatable classes are displayed in the right pane.

b To display the categories that can be assigned to service templates, click on the Example of custom templates link. The categories are displayed in the right pane.

c To display the template format information, click on the Format of the template link. The format information is displayed in the right pane.

d To display the sample templates, click on the Examples of custom templates link. The sample templates are displayed in the right pane.

Procedure 6-2 To convert standard service templates to XML API service templates

Perform this procedure to convert a service, site, L2 access interface, or SDP binding template created in 5620 SAM, Release 5.0 or earlier, to an XML API service template.

1 Choose Manage→ Templates from the 5620 SAM main menu. The Manage Templates form opens.

2 Click on the Old Templates button and choose one of the following:

- Browse/Convert Old Service Templates to convert existing service templates and child objects to XML API service templates
- Browse/Convert Old Mirror Service Templates to convert existing mirror service templates and child objects to XML API service templates

The Browse/Convert Old Service Templates form opens with the Service Template (Service Management) icon selected.

3 Click on the Search button. A list of service templates appear.

4 Perform one of the following.

a Select a service template and click on the Convert to XML API Template button. The selected template is converted to XML API based service template. After a successful conversion, the *Service_type* Service Template (Edit) form opens with the General tab displayed.

b Select several service templates and click on the Convert to XML API Template button. The selected templates are converted to XML API-based service templates. After a successful conversion, the newly created XML API templates are listed on the Manage Service Templates form. Click on the Cancel button to close the form. Go to step 6.

5 Perform one of the following.

a Click on the Cancel button to close the *Service_type* Service Template (Edit) form.

b Configure the XML API service template. Perform the following steps:

i Configure the parameters:

- Name
- Description
- Type
- Starting Point

ii Perform steps 14 to 20 of Procedure 6-3.

6 Close the Browse/Convert Old Service Templates form.

7 Close the Manage Service Templates form.

Procedure 6-3 To create an XML API template

Perform this procedure to create an XML API template for templatable service and tunnel objects.

1 Perform one of the following.

a Use the Manage Templates form. Choose Manage→Templates from the 5620 SAM main menu. The Manage Templates list form opens.

b Use the Script Manager. Perform the following steps.

i Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

ii Choose the Template (Scripting) object from the drop-down menu.

2 Click on the Create button. The Template (Create) form opens, with the General tab displayed.

3 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- State
- Command Type
- Mode

4 Click on the Select button beside the Templated Object Categories parameter. The Select Template Category form opens.

5 Use the check boxes to select one or more object categories and click on the OK button. The Select Template Category form closes and the Template (Create) form refreshes with the object category name.

6 Click on the Select button beside the Templated Object Class Name parameter. The Select Configured Class - Template form appears with a list of related object classes.

7 Select a templatable object class from the list and click on the OK button. The Select Configured Class - Template form closes and the Template (Create) form refreshes with the templatable object class name.

See Procedure 6-1 for information about listing the sample templates.

8 Configure the Generate First (Base) Version parameter.

Note — Alcatel-Lucent recommends that you enable the Generate First (Base) Version parameter and you select only the minimum required set of templated object properties by clicking on the Templated Object Attributes button. Templated attributes can be added or removed at any time using the Script Editor Property Selector form.

9 If you enabled the Generate First (Base) Version parameter in step 8, the Select Templated Object Attributes button becomes active. Go to step 10. Otherwise, go to step 12.

10 Perform one of the following.

a Click on the Select Templated Object Attributes button. The Select Templated Object Attributes To Be Included form opens. Use this form to select tabs, groups or fields you want to include in the generated template. Go to step 11 for configuration information.

b Go to step 12.

11 Specify the objects to be included in the template:

a On the left side of the form, configure the product compatibility by specifying the node type and version number to which you want the template to apply. Click on the All button at the top of the list to enable all the entries in the list. By default, none of the items are initially selected. The selections that you make on this side of the form affect the groups and field selections that you can make on the right side of the form.

b On the right side of the form, choose the groups and fields to include in the generated version of the template. By default, all of the items are initially selected. Deselect the All button at the top of the list to deselect all the entries in the list. You can then choose the required entries individually.

c Click on the OK button to apply the changes. The Select Templated Object Attributes To Be Included form closes.

12 Perform one of the following.

a Click on the OK button to complete the creation of the XML API template without creating or binding child templates. The Template (Create) form closes. Go to step 20.

Note — If you enabled the Generate First (Base) Version parameter in step 8, the 5620 SAM generates a script version with default Velocity properties, including a default Velocity UI header and an XML API configuration script. See step 16 for information about how to view and modify the generated script version.

b Click on the Apply button. A dialog box appears.

13 Click on the OK button. A Template (Edit) form opens.

14 Click on the Components tab button.

15 Perform one of the following.

a Create an XML API template for a child object. The XML API template for the child object is automatically bound with the parent template. Perform the following steps:

i Select and right-click on the *Object_type* Template icon and choose Create Child Template from the contextual menu. The Template (Create) form opens for the child object.

ii Perform steps 3 to 11.

Note — To group parent and child templates during a sort operation using the Manage Templates form, Alcatel-Lucent recommends that you configure the Description parameter for each child template using the value from the parent template.

iii Repeat step 15 to create additional templates for child objects of the parent object selected in step 4, if required.

iv Close the Template (Create) form.

b Bind the template with an existing child XML API template. Perform the following steps:

i Select and right-click on the *Object_type* Template icon and choose Bind Existing Template from the contextual menu. The Select Template(s) To Create Binding form opens with a list of available child templates.

ii Choose a template from the list and click on the OK button. The Select Template(s) To Create Binding form closes.

16 Click on the Versions tab button and perform one of the following steps.

a Edit or preview the generated XML API configuration template script.

i Select the script version and click on the Properties button. The Script Editor form opens.

Note — You can also click on the Script Editor button from any tab to open the Script Editor form with the latest version of the XML API script.

ii You must edit the script to suit your template requirements, which includes modifying NE-specific properties and attributes. See Procedures 4-5 to 4-6 in chapter 4 for more information about modifying the Velocity UI header.

See section "Velocity overview" in chapter 3 for more information about Velocity properties.

See section "Velocity GUI Builder" in chapter 3 for more information about how to modify the generated UI using the 5620 SAM GUI builder.

Note — The Velocity UI header and XML API configuration script are generated only if you enabled the Generate First (Base) Version parameter in step 8.

iii Close the Script Editor.

b If you did not enable the Generate First (Base) Version parameter in step 8, manually enter an XML API configuration template script.

i Click on the Create button. A blank script editor form opens.

ii Create the XML API script. See section "XML API script syntax" in chapter 4 for more information.

iii Save the script and close the Script Editor.

17 Click on the Spans tab button. Perform one of the following.

a View the existing span or spans of control applied to the script.

i Select a span from the list.

ii Click on the Properties button. The Span – *span_name* (Edit) form opens.

iii Click on the following tab buttons:

- Contents to view the scripts to which the span of control is applied, or add a script to the span of control
- Span of Control Profiles to view the span of control profiles that are applied to the span of control
- User Groups to view the user groups to which the span of control profiles are associated
- Users to view the users that belong to the user groups

iv Click on the Cancel button to close the form.

v Go to step 20.

b Apply a span of control to the script.

i Click on Add button. The Select Span(s) – Script form opens.

ii Select the span or spans of control that you want to add to the script.

iii Click on the OK button. The Select Span(s) – Script form closes and a dialog box appears.

iv Click on the OK button to confirm the action.

18 Click on the Create Tunnel from Template button, if required.

19 Perform one of the following:

a To create a dynamic LSP from a tunnel template.

b To create an LSP Path from a tunnel template.

c To create an SDP from a tunnel template.

See the *5620 SAM User Guide* for more information.

20 Close the Template (Edit) form.

Procedure 6-4 To create an XML API service template from an existing network object

1 Choose Manage→Services from the 5620 SAM main menu. The Manage Service form opens.

2 Specify a filter to search for existing templatable service objects and click on the Search button.

3 Choose an entry from the list and click on the Properties button. The *Service_object* (Edit) form opens.

4 Click on the Create Template button. The Service Template (Create) form opens.

5 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- State
- Command Type
- Mode
- Generate Velocity Properties

Note — Alcatel-Lucent recommends that you enable the Generate Velocity Properties parameter and you select only the minimum required set of templated object properties by clicking on the Templated Object Attributes button. Templated attributes can be added or removed at any time using the Script Editor Property Selector form.

6 If you enabled the Generate Velocity Properties parameter in step 5, the Select Templated Object Attributes button becomes active. Go to step 7. Otherwise, go to step 9.

7 Perform one of the following.

a Click on the Select Templated Object Attributes button. The Select Templated Object Attributes To Be Included form opens. Use this form to select tabs, groups or fields you want to include in the generated template. Go to step 8 for configuration information.

b Go to step 9.

8 Configure the required groups and fields:

a Specify the groups and fields to include in the generated version of the template. By default, all of the items are initially selected. Deselect the All button at the top of the list to deselect all the entries in the list. You can then choose the required entries individually.

b Click on the OK button to apply the changes. The Select Templated Object Attributes To Be Included form closes.

9 Perform one of the following.

a Click on the Do not include contained objects radio button to specify that children objects of the selected instance are not to be included in the configuration template. This option is selected by default.

b Click on the Include contained objects radio button to specify that children objects of the selected instance are to be included in the configuration template.

Note — If you select the Include contained objects option, in addition to enabling the Generate Velocity Properties parameter, multiple templates and bindings are created—one template and binding for each contained network object. The 5620 SAM generates a script version for each template with a Velocity UI header and an XML API configuration script. The default values of Velocity properties are based on the actual configured parameter values of the managed network object.

If you do not enable the Generate Velocity Properties parameter, a Velocity UI header is not generated for the script. However, an XML API configuration script, without Velocity properties, is generated. A single template is created with a script that includes only the XML API configuration script, whether you select the Include contained objects option. The property values are based on the actual configured parameter values of the managed network object. You must manually add a Velocity UI header and properties using the Script Editor.

Alcatel-Lucent recommends that you disable the Generate Velocity Properties parameter if you want to write more complex templates that involve multiple managed object instances. For example, you can configure a single template that creates an Epipe VLL service, including two Epipe sites, and each site including one L2 access interface and one spoke SDP binding.

10 Click on the Apply button.

11 Perform steps 12 to 20 of Procedure 6-3.

12 Close the Manage Services form.

Procedure 6-5 To create an XML API tunnel template from a network object

Use this procedure to create a template from a network object such as a dynamic LSP, or LSP path. You can use the following procedures in the *5620 SAM User Guide* to create LSP variations:

- To create a dynamic LSP using a dynamic LSP tunnel template
- To create a LSP path using an LSP path tunnel template
- To create an SDP using an SDP tunnel template

1 Choose Manage→MPLS→Dynamic LSPs from the 5620 SAM main menu. The Manage Dynamic LSPs form opens.

2 Specify a filter to search for existing templatable LSP objects and click on the Search button.

3 Choose an entry from the list and click on the Properties button. The *Dynamic_LSP* (Edit) form opens.

4 Click on the Create Template button. The Template (Create) form opens with the General tab displayed.

5 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- State
- Command Type
- Mode
- Generate Velocity Properties

Note — Alcatel-Lucent recommends that you enable the Generate Velocity Properties parameter and you select only the minimum required set of templated object properties by clicking on the Templated Object Attributes button. Templated attributes can be added or removed at any time using the Script Editor Property Selector form.

6 If you enabled the Generate Velocity Properties parameter in step 5, the Select Templated Object Attributes button becomes active. Go to step 7. Otherwise, go to step 9.

7 Perform one of the following.

a Click on the Select Templated Object Attributes button. The Select Templated Object Attributes To Be Included form opens. Use this form to select tabs, groups or fields you want to include in the generated template. Go to step 8 for configuration information.

b Go to step 9.

8 Configure the required groups and fields as follows:

a Select the groups and fields to include in the generated version of the template. By default, all of the items are initially selected. De-selecting the All button at the top of the list de-selects all of the entries in the list. You can then chose the required entries individually.

b Click on the OK button to apply the changes. The Select Templated Object Attributes To Be Included form closes.

9 Perform one of the following.

a Click on the Do not include contained objects radio button to specify that child objects of the selected instance are not included in the configuration template. This option is selected by default.

b Click on the Include contained objects radio button to specify that children objects of the selected instance are included in the configuration template.

Note — If you select the Include contained objects option, and enable the Generate Velocity Properties parameter, one template and binding is created for each contained network object. The 5620 SAM generates a script version for each template with a Velocity UI header and an XML API configuration script. The default values of Velocity properties are based on the configured parameter values of the managed network object.

If you disable the Generate Velocity Properties parameter, a Velocity UI header is not generated for the script. However, an XML API configuration script, without Velocity properties, is generated. One template is created with a script that includes only the XML API configuration script, whether you select the Include Contained Objects option. The property values are based on the configured parameter values of the managed network object. You must manually add a Velocity UI header and properties using the Script Editor.

Alcatel-Lucent recommends that you disable the Generate Velocity Properties parameter if you need to write complex templates that require multiple managed object instances.

10 Click on the Apply button. A dialog box appears.

11 Click on the OK button. The Template (Create) form closes and the Template (Edit) form opens.

12 Perform steps 12 to 20 of Procedure 6-3.

Procedure 6-6 To create an XML API template from an example

Use this procedure to access examples of template configuration. The examples provide a way for users to learn the procedures, practices and concepts that are required to configure XML API templates; for example, creating LSP templates used by auto-tunnel rules.

1 Perform one of the following.

   a Use the Manage Templates form. Choose Manage→Templates from the 5620 SAM main menu. The Manage Templates list form opens.

   b Use the Script Manager. Perform the following steps.

      i Choose Tools→Scripts from the 5620 SAM main menu. The Script Manager list form opens.

      ii Choose the Template (Scripting) object from the drop-down menu.

2 Click on the Browse Examples button. The Browse Template Examples form opens.

3 From the tree, locate a sample script.

4 Select the sample script.

5 Click on the Description tab button to view a description of the sample script.

6 Click on the Select Format and Range Policies tab button, if applicable. See the *5620 SAM User Guide* for more information about configuring range and format policies.

7 Click on the Preview tab button to view the sample script.

8 Click on the Create Script button to create an instance of the sample script. The Template (Create from example) form opens.

9 Configure the parameters:

- Auto-Assign ID
- Script ID
- Name
- Description
- Type
- Starting Point
- Command Type
- Mode

10 Click OK. Template (Create from example) form closes.

11 Click on the Close button. The Browse Examples of Scripts form closes.

12 Close the Script Manager list form.

Procedure 6-7 To create a network object from an XML API service template

1 Choose Create→Service From Templates from the 5620 SAM main menu. The Create Service From Template form opens.

2 Select a service template and click on the OK button.

3 Configure the parameters, as required. The parameters and tab buttons that appear depend on the object that you are creating and the Velocity UI header that you have defined for the XML API template script.

4 Configure the Show created object parameter.

5 Click on the OK button to run the XML API configuration script and create the service object. Any errors that are generated are shown at the bottom of the form in the XML results. If there are no errors and if you enabled the Show created object parameter in step 4, the Create *Service* From Template form closes and the service object property form opens.

6 Click on the Components tab button.

7 Right-click on the parent object in the components tree and choose Create Site From Template from the contextual menu.

Note — The Create Site From Template contextual menu option is enabled only if one or more site templates exist and are bound to the service template.

8 Choose a site template from the contextual menu. The Create Service Site From Template form opens.

9 Configure the parameters, as required. The parameters and tab buttons that appear depend on the object that you are creating and the Velocity UI header that you have defined for the XML API template script.

10 Users who are assigned the template script management scope of command role can preview the script for debugging purposes by performing the following:

   i Click on the Preview XML API Command button to preview the script for debugging purposes, if required.

   ii Close the Preview window.

11 Click on the OK button to run the XML API configuration script and create the service object. Any errors that are generated are shown at the bottom of the form in the XML results. If there are no errors and if you enabled the Show created object parameter in step 4, the Create *Service* Site From Template form closes and the service object property form opens.

12 Repeat steps 6 to 11 to create child objects from child templates that are bound to the site template, if required.

13 Close the service object property form.

Procedure 6-8 To modify a network object using an XML API service template

1 Choose a network object that has a template associated with it. For example, a VPLS service that was previously created from a template.

Note — Every network object that is templatable has an associated template listed on the Template tab of the object configuration form. Only one network object can be updated using a template at a time.

2 Click on the Update Using Template button. A menu listing associated modification templates will open.

3 Select one of the modification templates. The Modify Object Using Template form opens.

Note — The form will be updated to reflect changes in the network object. Any value that has been changed on the network object will be overwritten by the value defined by the template.

4 Configure the parameters, as required. The parameters and tab buttons that appear depend on the object that you are creating and the Velocity UI header that you have defined for the XML API template script.

5 Users who are assigned the template script management scope of command role can preview the script for debugging purposes by performing the following:

- **i** Click on the Preview XML API Command button to preview the script for debugging purposes, if required.
- **ii** Close the Preview window.

6 Click on the OK button to apply the modifications to the network object.

6.6 Format and range policies configuration of services and LSPs using templates

The 5620 SAM supports both format and range policies. Format policies manage how services, LSPs, L2 and L3 access interfaces are named and described. Range policies manage the ID numbers that are assigned to services, LSPs, L2 and L3 access interfaces. For example, you can configure a range policy to specify a range of 200 and 499 for all IDs for a service. You can configure a format policy to specify that service names do not exceed 10 characters. See chapter 60 of the *5620 SAM User Guide* for more information on format and range policy configuration.

The 5620 SAM allows an operator to apply pre-configured format and range policies to LSPs and services. The following section describes how a template designer can create a template to be used when creating LSPs and services by linking to existing format and range policies.

Format policy

When creating a service or LSP using the templates GUI Builder and Script Editor, you can provide a hard link to the format policy that you want to associate to that service or LSP. In this case, it does not matter if the policy is associated to the user, it will be used anyway. The following code shows how this is applied on the Service Name property.

```
<displayedName
formatPolicy="pvp:format-1">$displayedName</displayedName>
```

where pvp:format-1 is the existing formatPolicy

Or you could use policy auto-selection where the server automatically selects the best policy which is based on the user or group and priority. The following code shows how this is applied on the Description property.

```
<description formatPolicy="auto">$description</description>
```

Another option for implementing the hard link format rule is to allow auto-filled text only. For example, not allowing the user to enter text. The following code shows an example where the policy “pvp:format-1” will be used to specify the displayedName.

```
<displayedName formatPolicy="pvp:format-1"/>
```

The previous two methods described how a template designer could apply a format policy using one entry only. Code 6-3 shows how multiple parameters, in this case two text entries, can be entered by the user into the displayedName field.

Code 6-3: Sample code showing how multiple parameters can be entered into the displayedName field

```
##[auto-filled text] $displayedName1 [auto-filled text]
$displayedName2 ...
        <propertyValuePolicyMap>
          <item>
            <key>displayedName</key>
            <value>
              <generic.PropertyValueEntry>
                <policyFullName>auto</policyFullName>
                <parameters>
                  <string>$displayedName1</string>
                  <string>$displayedName2</string>
                </parameters>
              </generic.PropertyValueEntry>
            </value>
          </item>
        </propertyValuePolicyMap>
```

Range policy

You can implement a range policy against an existing service using templates. The following code example shows how a policy (pvp:range-1) is selected using the GUI Builder and Script Editor. It is similar to the method used for format policy.

```
<serviceId rangePolicy="pvp:range-1"/>$serviceId</serviceId>
```

You can also use policy auto-selection for range policies, as shown in the following code example.

```
<serviceId rangePolicy="auto">0</serviceId>
```

Appendices

A. Template management examples

A.1 Template management examples overview

Using the 5620 SAM script management examples can help template developers create, manage, and execute 5620 SAM XML API templates.

To view the examples, choose Manage→Templates to open the Manage Templates form, and then click on the Browse Examples button to open the Browse Template Examples form.

The XML API configuration templates simplify tunnel and service creation by reducing the steps required to create a tunnel or service component. The sample templates contain examples for both tunnel and service templates.

A.2 Auto-tunnel

Auto-tunnel templates may have to be modified before they are selected in an auto-tunnel rule. The modifications include retaining all auto-tunnel controlled variables in the template script.

The template examples are:

- Dynamic LSP for auto-tunnel rules with format policies
- LSP path for auto-tunnel rules
- Service tunnel for auto-tunnel rules with simplified format policies

Dynamic LSP for auto-tunnel rules with format policies

This template is an example of an MPLS LSP template used by auto-tunnel rules. The template demonstrates how to create an LSP tunnel by allowing the user to enter the source site ID, destination site ID and destination IP address. This template also demonstrates the use of format policies to build a custom name and description of the LSP.

When modifying an LSP template for an auto-tunnel rule, make sure the following variable tags are present in the template; everything else in the template must be constant:

```
<distinguishedName>$parent.getObjectFullName()</distinguishedName>

<sourceNodeId>$sourceNodeId</sourceNodeId>

<pathId>$pathId</pathId>

<displayedName>$displayedName</displayedName>

<destinationIpAddress>$destinationIpAddress</destinationIpAddress>

<destinationNodeId>$destinationNodeId</destinationNodeId>
```

LSP path for auto-tunnel rules

This template is an example of an MPLS LSP path template. This template demonstrates how to create an LSP path by first selecting the LSP and then entering a few basic LSP path attributes.

When modifying an LSP template for an auto-tunnel rule, make sure the following variable tags are present in the template; everything else in the template must be constant:

```
<distinguishedName>$parent.getObjectFullName()</distinguishedName>

<resourceId>$resourceId</resourceId>
```

Service tunnel for auto-tunnel rules with simplified format policies

This template is an example of an SDP template used by auto-tunnel rules. The template demonstrates how to create a service tunnel by allowing the user to enter basic identification information including type of underlying transport, MPLS signaling on or off, signaling type for inner labels, source site, destination site, and administrative state. This template also demonstrates the use of format policies to build a custom name and description of the SDP.

When modifying an SDP template for an auto-tunnel rule, make sure the following variable tags are present in the template; everything else in the template must be constant:

```
<distinguishedName>$parent.getObjectFullName()</distinguishedName>

<sourceNodeId>$sourceNodeId</sourceNodeId>

<destinationNodeId>$destinationNodeId</destinationNodeId>

<displayedName>$displayedName</displayedName>

<ldpEnabled>$ldpEnabled</ldpEnabled>

<underlyingTransport>$underlyingTransport</underlyingTransport>

<pathId>$pathId</pathId>
```

A.3 Format and range policy

The template example is:

- Service template with format and range policies

Service template with format and range policies

This template is an example of a service template using format and range policies. The template is used to create a sample VPLS service. It demonstrates different options for format and range policies from templates.

A.4 Services

The services templates are simplified templates for creating basic service objects. For each service type, there are separate scripts for creating the service, the service site, and service SAP.

There are template examples for the following services:

- VPLS
- Epipe
- VPRN
- IES

VPLS

The following sections describe how to configure VPLS services, sites, SAPs, I-sites, and I-SAPs.

VPLS service

To use the VPLS service template, you must import this service template, the VPLS site template, and the VPLS SAP template.

If creating an I-VPLS template, you must import the VPLS service template, the VPLS I-site template, and the VPLS I-SAP template.

You must then create the bindings within the template manager for this template.

The bindings between service, site, and SAP templates can be made within the components tab of the VPLS service template after the template have been imported.

They should be bound as shown:

```
VPLS Service
  |_VPLS Site
     |_VPLS SAP
```

or

```
VPLS Service
  |_VPLS I-Site
     |_VPLS I-SAP
```

VPLS site

For instructions on using the VPLS site template, see "VPLS service".

This template has a modified filter that is used when selecting a site. Only sites in that filter will be shown; this lets you specify node types that support VPLS. In this script, the 7750 SR, 7450 ESS, 7250 SAS, and 7710 SR are specified.

VPLS SAP

For instructions on using the VPLS SAP template, see "VPLS service".

VPLS SAP Modify

For instructions on using the VPLS SAP Modify template, see "VPLS service".

This template must be created with the Command Type parameter set to Modify.

VPLS I-site

For instructions on using the VPLS I-site template, see "VPLS service".

This template has a modified filter that is used when selecting a site. Only sites in that filter will be shown; this lets you specify node types that support VPLS. In this script, the 7750 SR, 7450 ESS, and 7710 SR are specified.

VPLS I-SAP

For instructions on using the VPLS I-SAP template, see "VPLS service".

Epipe

The following sections describe Epipe services, sites, and SAPs.

Epipe service

To use the Epipe service template, you must import this service template, the Epipe site template, and the Epipe SAP template.

You must then create the bindings within the template manager for this template.

The bindings between service, site, and SAP templates can be made within the components tab of the Epipe service template after the templates have been imported.

They should be bound as shown:

```
EPIPE Service
  |_EPIPE Site
     |_EPIPE SAP
```

Epipe site

For instructions on using the Epipe site template, see "Epipe service".

Epipe SAP

For instructions on using the Epipe SAP template, see "Epipe service".

Epipe SAP Modify

For instructions on using the Epipe SAP Modify template, see "Epipe service".

This template must be created with the Command Type parameter set to Modify.

VPRN

The following sections describe VPRN services, sites, and SAPs.

VPRN service

To use the VPRN service template, you must import this service template, the VPRN site template, and the VPRN SAP template.

You must then create the bindings within the template manager for this template.

The bindings between service, site, and SAP templates can be made from within the components tab of the VPRN service template after the templates have been imported.

They should be bound as shown:

```
VPRN Service

  |_VPRN Site

     |_VPRN SAP
```

VPRN site

For instructions on using the VPRN site template, see "VPRN service".

This template has a modified filter that is used when selecting a site. Only sites in that filter will be shown; this lets you specify node types that support VPRN. In this script, the 7750 SR and 7710 SR are specified.

VPRN SAP

For instructions on using the VPRN SAP template, see "VPRN service".

VPRN SAP Modify

For instructions on using the VPRN SAP Modify template, see "VPRN service".

This template must be created with the Command Type parameter set to Modify.

IES

The following sections describe IES services, sites, and SAPs.

IES service

To use the IES template, you must import this service template, the IES site template, and the IES SAP template.

You must then create the bindings within the template manager for this template.

The bindings between service, site, and SAP templates can be made within the components tab of the IES service template after the templates have been imported.

They should be bound as shown:

```
IES Service
  |_IES Site
     |_IES SAP
```

IES site

For instructions on using the IES site template, see "IES service".

This template has a modified filter that is used when selecting a site. Only sites in that filter will be shown; this lets you specify node types that support IES. In this script, the 7750 SR, 7450 ESS, and 7710 SR are specified.

IES SAP

For instructions on using the IES SAP template, see "IES service".

IES SAP Modify

For instructions on using the IES SAP Modify template, see "IES service".

This template must be created with the Command Type parameter set to Modify.

A.5 Service tunnels

There are template examples for the following:

- SDP-Create IP tunnel. This template can be used to create remote service tunnels between two NEs.
- LSP-Create dynamic LSP. This script shows a sample XML API Velocity UI that can be used to create a dynamic LSP between two NEs.
- LSP-Create LSP path. This script shows a sample XML API Velocity UI that can be used to create an LSP path between two NEs.

A.6 Auto-Bind

There are template examples for the following:

- VPRN Spoke SDP Binding — Example of a VPRN spoke SDP binding used during the service Auto-Bind functionality
- VPLS Mesh SDP Binding — Example of a VPLS mesh SDP binding used during the service Auto-Bind functionality

A.7 Secured VPN

There are template examples for the following:

- Secured Service Components
  - Secured Service — This is a stripped-down template for a VPRN service used as the Secured/Delivery Service when creating an IPSec VPN. To use this template, you must import this template and the Secured Service Site. You must then create bindings from within the template manager between this template and the Secured Service Site template.
  - Secured Service Site — This is the stripped-down VPRN Site template used as the Secured/Delivery Service Site when creating an IPSec VPN. The bindings between Secured Service (VPRN Service) and this template can be created from within the template manager.
- Delivery Service Components
  - Delivery Service — This is a stripped-down template for the IES or VPRN services that can be used as the Delivery Service when creating an IPSec VPN. To use this template, you must import this template and the Deliver Service Site for IPSec template. Bindings must be created then from within the template manager between this service and the Delivery Service Site for IPSec template.
  - Delivery Service Site — This is the stripped-down template for IES service site used along with the Deliver Service template when creating an IPSec VPN. Bindings must then be created between this template and Deliver Service template from within the template manager.

B. Script management examples

B.1 Script management examples overview

Using the 5620 SAM script management examples can help script developers create, edit, and manage CLI scripts, XML API scripts, and script bundles.

To view the examples, choose Tools →Scripts to open the Script Manager, and then click on the Browse Examples button to open the Browse Examples of Scripts form.

Note — For information about the template examples that are also available from the Browse Examples of Scripts form, see Appendix A.

This appendix describes the 5620 SAM script samples that are available for basic CLI, XML API scripts and script bundles.

B.2 CLI script examples

The following sections describe how to configure basic QoS on a 7250 SAS and list information for route tables.

Configure basic QoS on a 7250 SAS

This script shows a sample CLI Velocity UI that allows a user to do the following:

- configure the QoS priority value mapping
- specify a 7250 SAS interface on which they will configure the:
  - 802.1p priority level to untagged frames that arrive at the specified interface
  - traffic shaping to set the PIR for the transmit interface
  - tail-drop profile to be one of the three available congestion management algorithm profiles
- assign a QoS priority level per destination MAC address, per VLAN

List VPRN table

This script shows a sample CLI Velocity UI that lists the following:

- CLI summary view of the route table for a selected VPRN service site
- CLI detailed view of the route table for a selected VPRN service site

B.3 XML API script examples

The XML API script contains examples in the following areas:

- Miscellaneous: contains examples for creating an ACL IP filter policy, scheduler policy, IP interface, PIM interface using ISIS link or OSPF link, and creating BGP Peers with CPAA
- Services: contains examples for creating VPLS, Epipe, VPRN, and IES services, as well as modifying a VPLS SAP
- OAM: contains examples for creating ICMP and LSP tests

Miscellaneous

The following sections describe miscellaneous scripts.

ACL IP filter

This script shows a sample XML API Velocity UI that can be used to create an ACL IP filter policy on special addresses and application protocols.

Note 1 — A deny-all entry is added implicitly at the end of the policy and is not shown in the UI.

Note 2 — The Application Protocols tab allows any non-0.0.0.0 addresses to have an entry for each application protocol listed while 0.0.0.0 addresses are ignored.

Create scheduler policy

This script shows a sample XML API Velocity UI that can be used to create a scheduler policy containing up to four schedulers. It allows the assignment of parent schedulers and tiers as well.

Configure IP interface

This script shows a sample XML API Velocity UI that configures an IP interface (protocols, IP address) on a port. The script allows the user to specify if the far end should also be configured. In this sample case, the script uses the physical link on the local port to discover the far-end port.

Configure PIM interface using ISIS link

This script shows a sample XML API Velocity UI which will configure PIM on a selected interface. The script allows the user to select an admin domain, ISIS level, router within the admin domain, and the selected level, then select an ISIS link on the selected router.

The majority of the information is derived from the 5650 CPAM, therefore the 5650 CPAM must be installed and running with the 5620 SAM.

Configure PIM interface using OSPF link

This script shows a sample XML API Velocity UI which will configure PIM on a selected interface. The script allows the user to select an admin domain, OSPF area-id, router within the admin domain, and the selected level, then select an OSPF link on the selected router.

The majority of the information is derived from the 5650 CPAM, therefore the 5650 CPAM must be installed and running with the 5620 SAM.

Create BGP Peers with CPAA

This script shows a sample XML API Velocity UI which will configure BGP Peers between the selected CPAA and selected router(s). The user also selects the BGP Peer Group to be used based on the BGP Peer Groups configured on the CPAA.

A BGP Peer Group of the same name must exist on the target router(s).

Services

The following sections describe service scripts.

Create VPLS

This script shows a sample XML API Velocity UI that can be used to create a VPLS service with no site, or two sites with the option for SAPs.

Create Epipe

This script shows a sample XML API Velocity UI that can be used to create a VLL service with no site, or two sites with the option for SAPs.

Create VPRN

This script shows a sample XML API Velocity UI that can be used to create a VPRN service with no site, or two sites with the option for SAPs.

Create IES

This script shows a sample XML API Velocity UI that can be used to create an IES service with no site, or one site and an option to create a SAP.

When creating a SAP, the script allows the user to configure an interface name, IP address, port, and access ingress and egress policies.

Modify VPLS SAP

This script shows a sample XML API Velocity UI that allows the modification of an existing VPLS SAP.

The script requires the user to choose a VPLS service, followed by a site, and then a SAP.

Create optical transport service

This script shows a sample XML API Velocity UI that can be used to create an optical transport service with termination points on both sites.

OAM

The following sections describe OAM scripts.

Create ICMP tests

This script shows a sample XML API Velocity UI that can be used to create and execute either an ICMP ping or an ICMP trace test from the context of a VPRN.

Create LSP tests

This script shows a sample XML API Velocity UI that can be used to create and execute either an LSP ping or an LSP trace test on multiple dynamic LSPs that originate from a site.

B.4 Script bundle examples

This section contains information about the e following 5620 SAM script bundle examples:

- Equipment Provisioning Bundle
- Network Protocol Interface Bundle
- Service Tunnels Bundle
- VPLS Service Bundle
- Epipe Service Bundle
- Service OAM Bundle
- IP Backhaul Auto-Provisioning Bundle
- 9500 Cpipe Service Bundle
- Channel-To-Channel (7705-7750) Multi-Service Bundle
- Uni-directional (7705-7750) Ethernet OAM Bundle

Note 1 — In each example, a control script name ends with CTL, an XML API script name ends with XML, and a template name ends with TMPL.

Note 2 — The first script that is executed in each example is a starting point script. Other starting point scripts that are executed later in an example are explicitly identified.

Equipment Provisioning Bundle

The Equipment Provisioning Bundle example uses control scripts to invoke CTL and XML API scripts that perform the following operations in sequence:

1 Create an IOM.

2 Create an MDA.

3 Configure network port parameters.

4 Configure access port parameters.

5 Optionally configure network interfaces and protocols.

Note — These operations are to be performed on a 7750 SR or a 7705 SAR. The equipment modifications are applied only if appropriate for the device type.

Dependencies

The Network Protocol Interface Bundle must be installed.

Execution flow

The following is the execution flow of the scripts in the Equipment Provisioning Bundle:

```
Equipment Provisioning CTL

↓

→ Configure IOM CTL

      → Configure IOM XML

↓

→ Configure Daughter Card CTL

      → Configure Daughter Card XML

↓

→ Configure Port CTL

      → Configure Ethernet Port XML

↓

→ Network Interface and Protocols CTL (from Network Protocol
Interface Bundle)
```

Network Protocol Interface Bundle

The Network Protocol Interface Bundle example uses control scripts to invoke CTL and XML API scripts that perform the following operations in sequence:

1 Enable network protocols.

2 Create a network interface.

3 Create a network protocol interface.

4 Create an OSPF area interface.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the Network Protocol Interface Bundle:

```
Network Interface and Protocols CTL
↓
→ Configure Protocols CTL
      → Configure Protocols XML
↓
→ Create Network Interface CTL
      → Create Network Interface XML
            → Configure Interface IP Address CTL
                  → Configure Interface IP Address XML
↓
→ Create Network Protocol Interfaces CTL
      → Create LDP Interface XML
      → Create MPLS Interface XML
      → Create RSVP Interface XML
↓
→ Create OSPF Area Interface CTL
      → Create OSPF Area XML
            → Create OSPF Interface CTL
                  → Create OSPF Interface XML
```

Service Tunnels Bundle

The Service Tunnels Bundle example uses control scripts to invoke CTL and XML API scripts and templates that perform the following operations in sequence:

1 Create one of the following between two NEs, at operator discretion:
- dynamic LSPs
- service tunnels
- dynamic LSPs and service tunnels

2 Create an Epipe service between the two NEs.

Note 1 — By default, a dynamic LSP uses an MPLS hopless path, which the 5620 SAM automatically creates if no other hopless path is available.

Note 2 — The created dynamic LSPs are bound to the created service tunnels.

Dependencies

The Epipe Service Bundle must be installed.

Execution flow

The following is the execution flow of the scripts in the Service Tunnels Bundle:

```
Create LSPs and Service Tunnels CTL
↓
→ Create MPLS Hopless Path XML
→ Create Dynamic LSP TMPL
→ Create Service Tunnels CTL
      → Create Service Tunnel TMPL
↓
→ Create Epipe Service CTL—from Epipe Service Bundle
```

VPLS Service Bundle

The VPLS Service Bundle example uses control scripts and templates to create a VPLS with multiple sites and SAPs.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the VPLS Service Bundle:

```
Create VPLS Service CTL
↓
→ Create VPLS Service TMPL
↓
→ Create VPLS Site CTL—opens GUI for input
      → Create VPLS Site TMPL
            → Create VPLS SAP CTL—opens GUI for input
                  → Create VPLS SAP TMPL
```

Epipe Service Bundle

The Epipe Service Bundle example uses control scripts and templates to create an Epipe service between two sites and optionally configure Ethernet OAM diagnostics for the Epipe service.

Dependencies

The Service OAM Bundle must be installed.

Execution flow

The following is the execution flow of the scripts in the Epipe Service Bundle:

```
Create Epipe Service CTL
↓
→ Create Epipe Service TMPL
      → Create Epipe Site CTL
            → Create Epipe Site TMPL
                  → Create Epipe SAP CTL
                        → Create Epipe SAP TMPL
↓
→ Ethernet Service OAM Configuration CTL—from Service OAM Bundle
```

Service OAM Bundle

The Service OAM Bundle example uses control scripts and XML API scripts to configure Ethernet OAM diagnostics on a service.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the Service OAM Bundle:

```
Ethernet Service OAM Configuration CTL
↓
→ Create Global MEG XML
↓
→ Enable CCM on Global MEG CTL
      → Create Remote MEP on Local MEG XML
      → Enable CCM on MEP XML
      → Create Test Suite and Generate Tests CTL
            → Create Test Suite XML
            → Generate Tests and Start Test Execution XML
```

IP Backhaul Auto-Provisioning Bundle

The IP Backhaul Auto-Provisioning Bundle example uses execution flows from other bundles to provision IP backhaul equipment, transport, network, service, and OAM objects on a 7705 SAR and a 7750 SR.

Dependencies

The following bundles must be installed:

- Equipment Provisioning Bundle
- Network Protocol Interface Bundle
- Service Tunnels Bundle
- Service OAM Bundle
- Epipe Service Bundle

Execution flow

The following is the execution flow of the scripts in the IP Backhaul Auto-Provisioning Bundle:

IP Backhaul Auto-Provision CTL—from IP Backhaul Auto-provisioning Bundle

↓

→ **Equipment Provisioning CTL—from Equipment Provisioning Bundle**

↓

→ **Network Interface and Protocols CTL—from Network Protocol Interface Bundle**

↓

→ **Create Service Tunnels CTL—from Service Tunnels Bundle**

9500 Cpipe Service Bundle

The 9500 Cpipe Service Bundle example uses control scripts and XML API scripts to create a 9500 MPR Cpipe service.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the 9500 Cpipe Service Bundle:

```
9500 Cpipe Service Configuration CTL
↓
→ Create 9500 Cpipe Service XML
↓
→ 9500 Cpipe SAP Configuration CTL—opens GUI for input
      → Create 9500 Cpipe SAP XML
```

Channel-To-Channel (7705-7750) Multi-Service Bundle

The Channel-To-Channel (7705-7750) Multi-Service Bundle performs the following sequence of actions.

1 Configure a DS0 channel group on a 7705 SAR ASAP port.

2 Configure a DS0 channel group on a a 7750 SR OC3 CES or OC3 ASAP card.

3 Create a VLL Apipe, Cpipe, or Ipipe service between the channel groups.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the Channel-To-Channel (7705-7750) Multi-Service Bundle:

```
Config (7705-7750) Channel Services CTL

↓

→ Config 7750 Ds0 Channel CTL

      → Config 7705 ASAP Port XML

      → Create 7705 Ds0 Group XML

      → Create 7705 Port Channel XML

↓

→ Config 7705 Ds0 Channel CTL

      → Config 7750 CES Port XML

      → Create 7750 Sts1 Channel XML

      → Create 7750 DS3E3 Channel XML

      → Create 7750 DS1E1 Channel XML

      → Create 7750 Ds0 Group XML

↓

→ Create (7705-7750) VLL Channel Services CTL

      → Create (7705-7750) Ipipe XML

      → Create (7705-7750) Apipe XML

      → Create (7705-7750) Cpipe XML
```

Uni-directional (7705-7750) Ethernet OAM Bundle

The Uni-directional (7705-7750) Ethernet OAM Bundle performs the following operations in sequence:

1 Create a Y.1731 global MEG for a VLL service between a 7705 SAR and a 7750 SR.

2 Create a unidirectional test that runs between the NEs.

Dependencies

This bundle has no dependencies.

Execution flow

The following is the execution flow of the scripts in the Uni-directional (7705-7750) Ethernet OAM Bundle:

Create ETH OAM Global MEG CTL

↓

→ **Create Ethernet CFM Global MEG XML**

↓

→ **Add (7705-7750) Services and CFM Tests to Global MEG CTL (starting point script)**

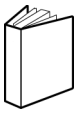
→ **Add (7705-7750) Service to Global Meg CTL**

→ **Add Service To Global MEG XML**

→ **Create (7705-7750) Ethernet CFM Tests CTL**

→ **Create Ethernet CFM Two Way Delay Test XML**

Customer documentation and product support



Customer documentation

http://www.alcatel-lucent.com/myaccess

Product manuals and documentation updates are available at alcatel-lucent.com. If you are a new user and require access to this service, please contact your Alcatel-Lucent sales representative.

Technical Support

http://support.alcatel-lucent.com

Documentation feedback

documentation.feedback@alcatel-lucent.com

3HE 06499 AAAF TQZZA Edition 01