Overview

Introduction

Cisco Convergent Network Solution Center: Subscriber Provisioning (CCNSC SP) is a GUI-based, data-driven provisioning system for subscriber Customer Premises Equipment (CPE) based upon the concept of service profiles, provisioning profiles, and task profiles. The purpose of this API is to enhance the CCNSC SP GUI capabilities by allowing a client to write in-house flow-through interface programs (scripts) to automate provisioning of the CPEs. The following topics are discussed in this chapter:

- “System Requirements”
- “Obtaining CCNSC SP API Development Support” section on page 1-2
- “Profile-based Provisioning” section on page 1-2
  - “Service Profile” section on page 1-3
  - “Provisioning Profile” section on page 1-4
  - “Task Profile” section on page 1-6
  - “Access to CORBA Objects” section on page 1-7.

System Requirements

The system requirements are:

- CCNSC SP Client—Solaris 2.7
- CCNSC SP Server:
  - Solaris 2.7
  - Oracle 8.1.6
  - Oracle Client (if Oracle is not running on the same server).

The programming environment requirements are:

- CORBA 2.1 compliant
- Integrated with IONA OrbixWeb 3.2 (Solaris 2.7, JDK1.2 is recommended) and Orbix 3.
Obtaining CCNSC SP API Development Support

Cisco has a new support program for developers who are enabling products with Cisco supported interfaces. The Developer Support Program is being developed to provide formalized support for Cisco interfaces to accelerate the delivery of compatible solutions to Cisco customers!

The Developer Support Program offers the following benefits:

- Minimal support fees
- Flexible support model—purchase support as needed or for a period of time
- Consistent level of support—defined problem priority and escalation guidelines
- Deliver products to market faster—dedicated program with interface experts to assist you.

To find out more about this program and obtain the Developer Support Agreement, visit our web site at http://www.cisco.com/go/developersupport. Upon receipt of your signed agreement, we will send you your contract ID number and instructions for opening support cases with Cisco Developer Support Engineers.

We look forward to working with you! Please don’t hesitate to contact us if you have further questions about this program.

Cisco Developer Support
developer-support@cisco.com
http://www.cisco.com/go/developersupport

Profile-based Provisioning

The flow-through APIs in CCNSC SP can be divided into several categories; Service Provisioning Manager (SPM_Server), Inventory Manager (INV_Server), Profile Analyzer (PA_Server), Profile Manager (PM_Server), and Service Template Manager (ServiceTempMgr). These APIs are provided as IDLs. Profile-based provisioning consists of five basic IDL modules to which you have access. Those modules are: ProfileManager, ProfileAnalyzer, ServiceRequestManager, InventoryManager, and ServiceTemplateManager, as shown in Figure 1-1.

Figure 1-1 CCNSC SP API Architecture
Service Profile

A service profile can be associated with provisioning profiles. It has service catalog dependent, but technology neutral definitions. A service profile describes the service offering, and it could be implemented by different network technologies. A provisioning profile describes the information needed to provision related devices in order to fulfill the service offering. A provisioning profile contains templates, default template data for each device type, and device model. It also contains other information such as the technology type and delivery method. The Profile Analyzer API allows a client to retrieve this information by passing appropriate filtering criteria. In order to make the IDL method generic, profile attributes are returned as a list of NVPair objects. It is therefore important to define the naming convention for these NVPair objects.

The attributes are described as follows:

- **SP_DESCRIPTION**—text description of the service offering, such as, a service package that can be ordered by the customer
- **SP_TYPE**—category of service definition, such as, “voice”, “primitive data”, “voice and data bundled”
- **SP_SUBTYPE**—further divides the service category into sub-categories (optional)
- **SP_CLASS_OF_SERVICE**—identifies the class of service (for example, platinum, gold, silver)
- **SP_SERVICE_FEATURES**—describes what constitutes a service offering (for example, a bundled voice and data package consisting of data channel and voice channel features). Service features are orderable units and can be provisioned incrementally.

Note: All service profile keyword attributes, with the exception of **SP_SERVICE_FEATURES**, start with **SP**. The **SP_SERVICE_FEATURES** attribute is a list of strings delimited by a comma.

Furthermore, service features may have inter-dependencies, for example:

- F-1—basic system configuration
- F-2—basic voice (dependent upon F-1)
- F-3—basic data (dependent upon F-1)
- F-4—voice channel (dependent upon F-2)
- F-5—data channel (dependent upon F-3).

The service dependency file defines feature dependencies to ensure system integrity.

Service Profile Example

Profile rules are described by a DTD file, which is part of the XML script language.

```
<!ELEMENT ServiceProfile(description,type,subType,classOfService,serviceFeature+)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT subType (#PCDATA)>
<!ELEMENT classOfService (#PCDATA)>
<!ELEMENT serviceFeature (#PCDATA)>
```
A profile example in XML follows:

```
<ServiceProfile>
  <description>Basic Data Service with IP connectivity and public IP address</description>
  <type>DataService</type>
  <classOfService>bronze</classOfService>
  <serviceFeature>NA</serviceFeature>
</ServiceProfile>
```

**Provisioning Profile**

A provisioning profile defines the mapping to network specifics for a service profile. It defines the technology used to implement a service profile, the network elements, and their configurations related to the network technology. All keyword attributes for a provisioning profile start with `PP`. Unlike service profile attributes, some attributes here are generated using variable values from the profile itself. The attribute keywords are described below:

- **PP_TECHNOLOGY**—technology type (for example, VoIP over ATM, VoIP over T1)
- **PP_$TYPE_$MODEL_SERVICE_FEATURES**
- **PP_$TYPE_$MODEL_DESTINATION**
- **PP_$TYPE_$MODEL_DOWNLOAD_METHOD**
- **PP_$TYPE_$MODEL_$SERVICEFEATURE_TEMPLATE_NAME**
- **PP_$TYPE_$MODEL_$SERVICEFEATURE_DEFAULT_TEMPLATE_DATA**
- **PP_TASK_PROFILE_NAME PP_TASK_LANGUAGE**

The Profile Analyzer also generates the following additional name-value pairs:

- **PP_$TYPE_SERVICE_FEATURES**
- **PP_$TYPE_DESTINATION**
- **PP_$TYPE_DOWNLOAD_METHOD**
- **PP_$TYPE_$SERVICEFEATURE_TEMPLATE_NAME**
- **PP_$TYPE_$SERVICEFEATURE_DEFAULT_TEMPLATE_DATA**

Basically, these new key words will not have the `$MODEL` token. `$TYPE` and `$MODEL` are variable values for the network element type and model. `$SERVICEFEATURE` is the service feature of a configuration template.

**Provisioning Profile Example**

Provisioning profile rules are in the DTD file listing that follows:

```
<!ELEMENT ProvisionProfile (Technology, NE+, Task+)>
<!ELEMENT Technology (#PCDATA)>
<!ELEMENT (Operation+,DeliveryMethod)>...
```

Basically, these new key words will not have the `$MODEL` token. `$TYPE` and `$MODEL` are variable values for the network element type and model. `$SERVICEFEATURE` is the service feature of a configuration template.
Below is a listing of a provisioning profile example:

```xml
<ProvisionProfile>
  <Technology>VoIPoPPP</Technology>
  <DefaultDownloadMethod>Telnet</DefaultDownloadMethod>
  <Destination>Running</Destination>
  <NetworkElement Type="CPE" Model="2421">
    <Operation Id="ADD">
      <ConfigTemplate ServiceFeature="BasicData">
        <TemplateName>CpeGroup:2421-voip-createData</TemplateName>
        <DefaultTemplateData>2421TemplateData</DefaultTemplateData>
      </ConfigTemplate>
    </Operation>
    <Operation Id="DELETE">
      <ConfigTemplate ServiceFeature="BasicData">
        <TemplateName>CpeGroup:2421-voip-deleteData</TemplateName>
        <DefaultTemplateData>2421Data</DefaultTemplateData>
      </ConfigTemplate>
    </Operation>
  </NetworkElement>
  <NetworkElement Type="Aggregator" Model="ESR10K">
    <Operation Id="ADD">
      <ConfigTemplate ServiceFeature="BasicData">
        <TemplateName>AggregatorGroup:esr10k-voip-createData</TemplateName>
        <DefaultTemplateData>esr10kData</DefaultTemplateData>
      </ConfigTemplate>
    </Operation>
    <Operation Id="DELETE">
      <ConfigTemplate ServiceFeature="BasicData">
        <TemplateName>AggregatorGroup:esr10k-voip-deleteData</TemplateName>
        <DefaultTemplateData>esr10kData</DefaultTemplateData>
      </ConfigTemplate>
    </Operation>
  </NetworkElement>
  <TaskProfile>
    <OperationId>ADD</OperationId>
    <TaskName>VoIPoPPPDataAddService</TaskName>
    <TaskLanguage>CCNSC_TDL_1.0</TaskLanguage>
  </TaskProfile>
  <TaskProfile>
    <OperationId>DELETE</OperationId>
    <TaskName>VoIPoPPPDataAddService</TaskName>
    <TaskLanguage>CCNSC_TDL_1.0</TaskLanguage>
  </TaskProfile>
</ProvisionProfile>
```
Task Profile

A task profile defines task sequences (operation flow) for Service Provisioning. There is only one task profile per operation (add/modify/delete service).

Task Profile Example

Contrary to service and provisioning profiles, a task profile is written in Tcl format. The example that follows shows how to add basic data service for VoIPoPPP.

```tcl
set TP_RESULT [ add_basic_data
    $SR_DEVICE_ID
    $PP_CPE_2421_BasicData_TEMPLATE_NAME
    $PP_CPE_2421_BasicData_DEFAULT_TEMPLATE_DATA
    TP_CONFIG
    TP_RESPONSE
if { $TP_STATUS != 0 }{
    ReturnVariables $TP_RESPONSE
}

set TP_STATUS [ deliver_config(
    $SR_DEVICE_ID
    $TP_CONFIG
    $PP_CPE_2421_DOWNLOAD_METHOD
    $PP_CPE_2421_DESTINATION
    TP_RESPONSE
if { $TP_STATUS != 0 }{
    ReturnVariables $TP_RESPONSE
}

set TP_STATUS [ get_associated_aggregator_id
    $SR_DEVICE_ID
    TP_AGGREGATOR_ID
    TP_RESPONSE
if { $TP_STATUS > 0 }{
    ReturnVariables $TP_RESPONSE
}

set TP_STATUS [ generate_config
    $TP_AGGREGATOR_ID
    $PP_Aggregator_ESR10K_BasicData_TEMPLATE_NAME
    $PP_Aggregator_ESR10K_BasicData_DEFAULT_TEMPLATE_DATA
    TP_CONFIG
    TP_RESPONSE
if { $TP_STATUS != 0 }{
    return $TP_RESPONSE
}

set TP_STATUS [ deliver_config(
    $TP_AGGREGATOR_ID
    $TP_CONFIG
    $PP_Aggregator_ESR10K_DOWNLOAD_METHOD
    $PP_Aggregator_ESR10K_DESTINATION
    TP_RESPONSE
return $TP_STATUS
```

Access to CORBA Objects

The flow-through APIs in CCNSC SP can be divided into several categories: Inventory Manager, Profile Manager, Profile Analyzer, Service Provisioning Manager, and Service Template Manager. These APIs are in the form of IDLs. You can access the CCNSC SP servers through the CORBA API. Example code on how to register with and exercise the servers is described in Chapter 2, “Creating a Client Application.”

Inventory Manager (INV_Server)

The Inventory Manager API (inv_mgr.idl) allows you to create a CPE, delete a CPE, and get CPE information. Prior to creating a CPE, a subnetwork or complex must exist. A CPE always resides within a subnetwork or complex. A CPE is identified by a fully qualified name and it is used during the service provisioning phase.

add_cpe()
Create a new CPE in the inventory.

delete_cpe()
Delete a CPE from the inventory.

get_cpe_info()
Retrieve information about a CPE from the inventory.

Profile Manager (PM_Server)

The Profile Manager manages three types of profiles (service, provisioning, and task). It allows the client to create, delete, or modify profiles and associate a service profile with one or more provisioning profiles.

associate()
Associate a service profile to a list of provisioning profiles.

create_group()
Create a new group.

create_profile()
Create a new profile.

de_associate()
Deassociate a set of profiles from the source.

delete_group()
Delete a group.
Profile-based Provisioning

**delete_profile()**
Delete all associations, then delete the profile.

**get_associated_profiles()**
Returns a set of associated profiles.

**get_profile()**
Retrieve profile type data.

**get_profile_list()**
List profile header information within a group.

**get_subgroups()**
Return a list of subgroups.

**modify_profile()**
Modify an existing profile.

**Profile Analyzer (PA_Server)**

A service profile, described by the `profile_analyzer.idl` file, can be associated with provisioning profiles. A service profile describes the service offering. A provisioning profile describes the information needed to provision related devices in order to fulfill the service offering. A provisioning profile contains templates, default template data for each device type, and the device model. It also contains other information such as the technology type and delivery method. The Profile Analyzer API allows a client to retrieve this information by passing appropriate filtering criteria.

**get_assoc_profile_attributes()**
Return associated profile attributes for a given service profile.

**get_service_profile_attributes()**
Return service profile attributes for a given service profile.

**Service Provisioning Manager (SPM_Server)**

The Service Provisioning Manager API is described by the `service_prov_mgr.idl` file. For information about the XML strings for T1, ATM, and Frame that are needed to process the service, refer to Appendix A, “Card-related XML String Parameters.”

**process_service_request()**
Process a service request.
Service Template Manager (ServiceTempMgr)

This API is contained in the `stm_mgr.idl` file. A template is a configuration file with configurable variables. Each configurable variable is called a template variable and is replaced with corresponding template data in order to generate a configuration file. The Service Template Manager provides the capability for a client to retrieve template variables and template data given a template name.

`get_template_data()`
Retrieve template attributes and data for a template.

`get_template_variables()`
Retrieve template variables.