Network Manager
4.2

Network Manager Discovery Collectors

Developer Guide

1.7

Note: Before using this information and the product it supports, read the information in "Notices" on page 105.

This edition applies to 4.2 of IBM Tivoli Network Manager and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2014, 2015.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this Guide.....v

Chapter 1: Collector Overview.....1

Collector Data.....2
- DISCO data (i.e. XML Schema defined fields).....2
- NCIM data (i.e. freeform data that mirrors the NCIM schema).....3
- Custom data (data lacking DISCO or NCIM support) 5

Writing a Collector.....7
- Sample collector code.....7

Migration Advice & Change Summary.....10
- DISCO Agent Change Summary.....10
- XML-RPC XML Interface Change Summary.....10

Chapter 2: XML Schema & XML-RPC Method Reference.....13

Mandatory RPC Method Reference.....13
- UpdateData().....13
- GetInfo().....14
- GetDeviceList().....16
- GetDeviceInfo().....17
- GetAssociatedAddresses().....18

Optional RPC Method Reference.....19
- GetInventory().....20
- GetEntities().....21
- GetLayer3Vpns().....24
- GetLayer2Vpns().....26
- GetMplsInterfaces().....28
- GetLayer3Connections().....29
- GetLayer2Connections().....30
- GetLayer1Connections().....31
- GetConnections().....32
- GetRanData().....34
- GetENodeBData().....34
- GetMmeData().....47
- GetSgwData().....56
- GetPgwData().....64
- GetHssData().....70
- GetPcrfData().....75
- GetEirData().....80

Extending The XML Schema.....84

Chapter 3: Network Manager Collector Support.....87

Collector Relevant Network Manager Processes.....87
- The Collector Finder.....90
- The XML-RPC Helper.....90
- The Collector Details Agent.....91
- The Collector Inventory Agent.....91
- The Collector Layer 2 Agent.....93
- The Collector Layer 3 Agent.....94
- The Collector VPN Agent.....94
- The Collector Layer 1 Agent.....95
- The Collector RAN Agent.....96

Collector-Relevant Network Manager Data Flow.....98
- Full Discovery Mode Stitching.....98
- Partial Rediscovery Stitching.....102

Notices.....105

Trademarks.....107

© Copyright IBM Corp. 2014, 2015

About this Guide

This guide provides the general information necessary to enable the development of Network Manager Collectors. It is externally supported by additional Java and Perl Collector support library documentation.
The guide breaks down into the following sections:

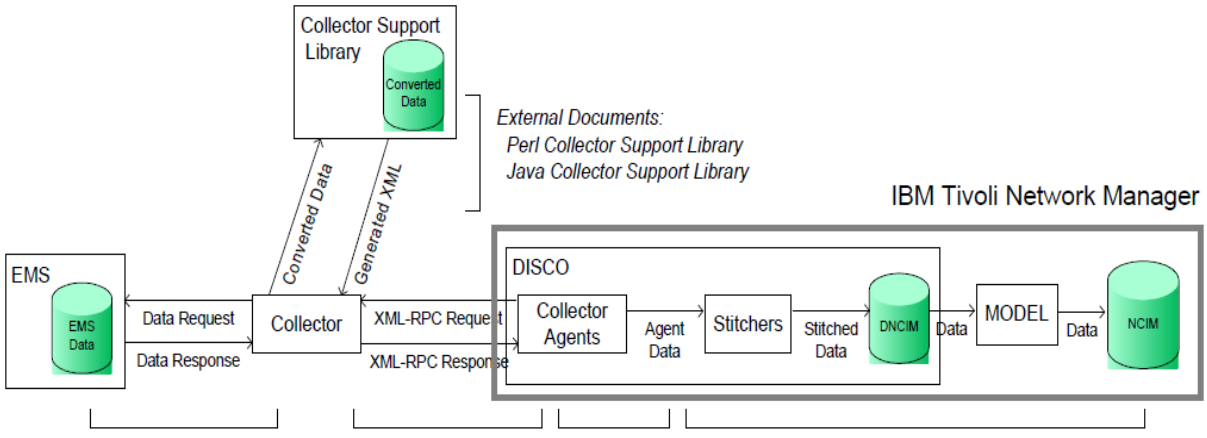
Chapter 1: Collector Overview introduces Collectors and how they fit in with the rest of Network Manager (high level).

Chapter 2: XML Schema & XML-RPC Method Reference provides reference information for the collector XML schema (in the rest of this document, XML schema), which governs the format collectors must use when sending data to DISCO in response to an XML-RPC call. This section also acts as an XML-RPC method reference(the methods DISCO calls to retrieve data).

Chapter 3: Network Manager Collector Support provides an insight into what Network Manager does with collector data. This information is useful both when deciding which features to add to a collector and during issue investigation.

Information regarding developing Perl and Java collectors can be found in the separate Perl and Java collector support library documentation. Note: The API documentation is generated from the Perl and Java library source code and as such follows the documentation style of commonly used for APIs written in that programming language.

Below is a data flow oriented chapter index to aid quick reference navigation of this document for those already familiar with the basics. Readers new to collectors are, however, encouraged to read through the document in order.



© Copyright IBM Corp. 2014, 2015

Chapter 1: Collector Overview

Collectors provide the solution to the problem of how to import data from an external data source into Network Manager at a level equivalent to the standard SNMP based agents. The collector system presents a clearly defined XML interface into Network Manager which allows integrations without requiring changes to the core Network Manager product.

Collectors retrieve data from a data source (e.g. an EMS) and make it available in a standard form (XML adhering to the XML schema specified in this document) via a defined interface (XML-RPC) that DISCO can use to import the data at agent level. There are a number of standard collector discovery agents that will import this data and pass it on to the discovery stitchers for processing (e.g. topology creation).

A collector is relatively independent of the rest of Network Manager; it is configured and run independently, perhaps even on a remote box. From a Network Manager point of view it is similar to any other discoverable network device the difference being that a single collector may provide data on many network devices.

Collectors are responsible for:

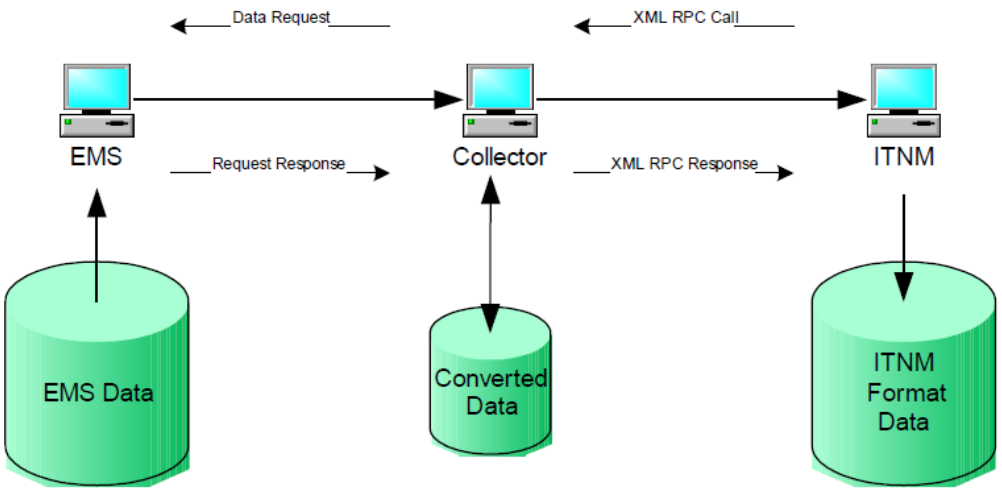
- Importing data from a data source
- Implementing an XML-RPC server to which DISCO can connect
- Implementing at a minimum those RPC methods marked as essential, as defined in Chapter 2: XML Schema & XML-RPC Method Reference
- Ensuring the data returned by these methods is of the correct format, as defined in Chapter 2: XML Schema & XML-RPC Method Reference

Network Manager does not impose limits on the choice of development language. However, Perl and Java modules have been developed that will aid development of Perl and Java based collectors.

The following figure shows how the collector interacts with the rest of Network Manager and the target EMS. Note that, although the collector-to-DISCO communications must be via XML-RPC, there are no such limitations on the collector-to-EMS communications method.

DISCO knows the collector exists because you configure DISCO with a seed host and port before running a discovery (where the host is the box running the collector, and the port is the port on which the collector is listening for DISCO XML-RPC requests).

© Copyright IBM Corp. 2014, 2015



Collector Data

From the Network Manager perspective collector data (the data in the XML-RPC message sent to Network Manager) falls roughly into three categories:

- Network Manager discovery process (DISCO) data
- NCIM data
- Custom data

These categories are simply a reflection of the name and position of the data in the XML-RPC response, and this name/position impacts whether the data is used by the DISCO processing stitchers, and how it travels to NCIM (if at all).

DISCO data (i.e. XML Schema defined fields)

These are data fields that may be used by DISCO to deduce the entities and relationships in the network and are the fields that are explicitly defined in the XML schema. DISCO fields can be (and often are) stored directly or indirectly in an associated NCIM field.

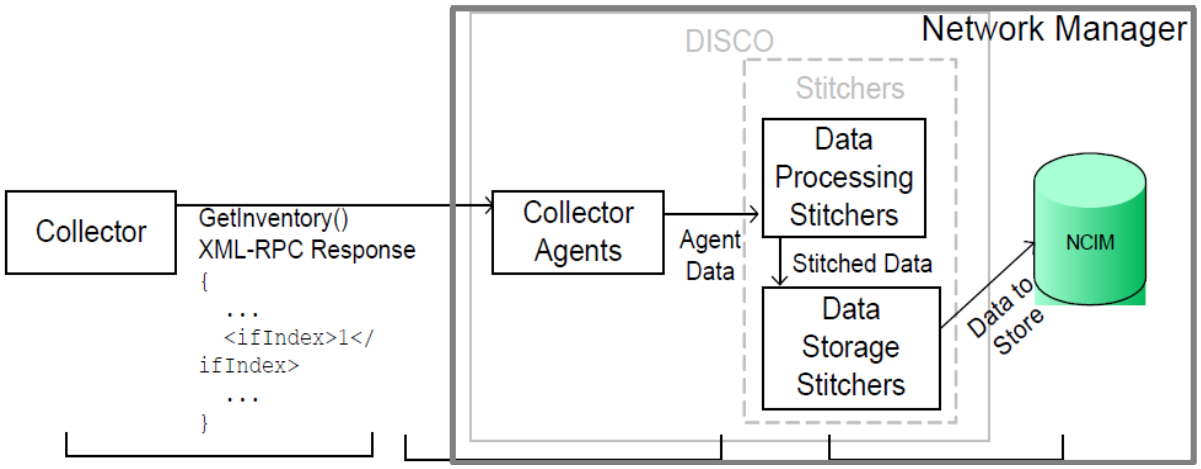
Put DISCO data in an XML-RPC response (i.e. populate non-freeform XML Schema tags) to take advantage of DISCO's data processing stitchers or to meet any mandatory data requirements of the XML RPC schema (the mandatory fields exsist because they are required in mandatory DISCO data processing).

For historical reasons the name used in the XML tag differs from DISCO and NCIM field names; this makes it difficult to follow the data flow during issue investigation. For example, the 'card' tag in the XML Schema becomes 'm_LocalNbrCard' in DISCO and changes again to 'cardNumber' in NCIM. Refer to chapter 3 for details as to how XML Schema tags map to DISCO field names and refer to the DNCIM stitchers ($PRECISION_HOME/disco/stitchers/DNCIM/) and $NCHOME/etc/precision/ModelNcimDb.cfg to see how DISCO fields relate to NCIM fields.

Any desired DISCO fields that are NOT represented in the published XML schema, for example, DISCO's m_LowerNeighbors field,which can be used in defining interface hierarchy, must to be passed as freeform data (see Extending The XML Schema for details on how to do this). This is an advanced usage of the collectors and

requires advanced knowledge of how and why DISCO uses data fields, which is out of the scope of this document.

The Figure below shows an example of DISCO data being processed by collectors and Network Manager.



Category: DISCO data
The ifIndex above is considered DISCO data because it is one of the tags explicitly defined in the XML Schema, which implies that the field produced by the Agents from the XML tag will be used in DISCOs data processing stitchers

XML Tag → DISCO Field
Section 3.1 provides information on the mapping of tags to Agent (DISCO) fields.

DISCO Field → NCIM Field
The stitchers in $PRECISION_HOME/disco/stitchers / define the mapping between DISCO field name and NCIM field name.
In this case m_IfIndex maps to networkInterface->ifIndex

NCIM data (i.e. freeform data that mirrors the NCIM schema)

Although DISCO data fields can end up in NCIM there is a clearer way to specify fields intended for NCIM only and this is where the NCIM data category comes in.

Fields considered 'NCIM data' are data fields that are not normally examined by the DISCO processing stitchers (i.e. not used by the existing stitching to deduce relationships): These fields are simply copied from the collector XML response to NCIM (while taking advantage of non-field specific features of the DISCO system).

NCIM style data fields can either be piggy-backed on to responses from standard XML-RPC calls via the freeform (extrainfo) sections of the collector responses, or they can be sent as the responses from custom XML-RPC based calls (i.e. XML-RPC added by users that are not supported by Network Manager out of the box and are intended to be called by custom Perl agents).

Either way the data for a table would be of a the following form;

```
<NCIMTableName>
    <NCIMFieldName>Value</NCIMFieldName>
    ...
</NCIMTableName>
```

It is your responsibility to ensure that the table names and field names match a real NCIM table. For information on the the NCIM schema, see the IBM Tivoli Network Manager Topology Database Reference.

NCIM data fields are stored in NCIM after any required translations (for example, of enumeration types from integer to text string) by DISCO's Data Storage stitchers (also known as DNCIM stitchers).

You must ensure that NCIM data is returned in response to an appropriate XML-RPC call (see Limitations below).

Limitations:

- Not all NCIM tables are currently supported.

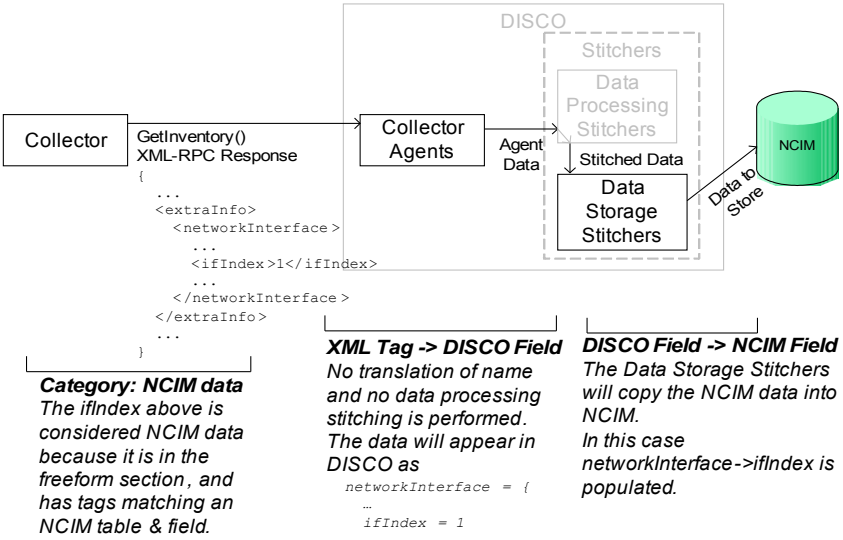
The NCIM tables that can currently be populated via piggy-backing freeform data to existing XML-RPC calls are as follows;

NCIM Table	**XML RPC Call** (in which NCIM data should be sent by collector)
physicalChassis	GetDeviceInfo() or GetEntities() for Chassis entities
snmpSystem	GetDeviceInfo() or GetEntities() for Chassis entities
computerSystem	GetDeviceInfo() or GetEntities() for Chassis entities
operatingSystem	GetDeviceInfo() or GetEntities() for Chassis entities
networkInterface	GetInventory() or GetEntities() for port entities.
physicalConnector	GetInventory() or GetEntities() for port entities.
transmissionTp	GetInventory() or GetEntities() for port entities.
physicalSlot	GetEntities() for slot entities
physicalBackplane	GetEntities() for backplane entities
physicalPowerSupply	GetEntities() for PSU entities
physicalFan	GetEntities() for fan entities
physicalSensor	GetEntities() for sensor entities
physicalOther	GetEntities() for other/unknown entities
physicalCard	GetEntities() for card (module) entities

- Only a single entry per NCIM table

You can only specify a single instance of a table in the XML response. This is not usually an issue for the currently supported NCIM tables as a separate freeform data section can be returned within each entity due to the way in which stitchers work.

- If, for some reason, a collector responds with a DISCO data field and NCIM data field both destined for the same NCIM table field then the NCIM style field will be used in NCIM population and the DISCO field used during discovery stitching.

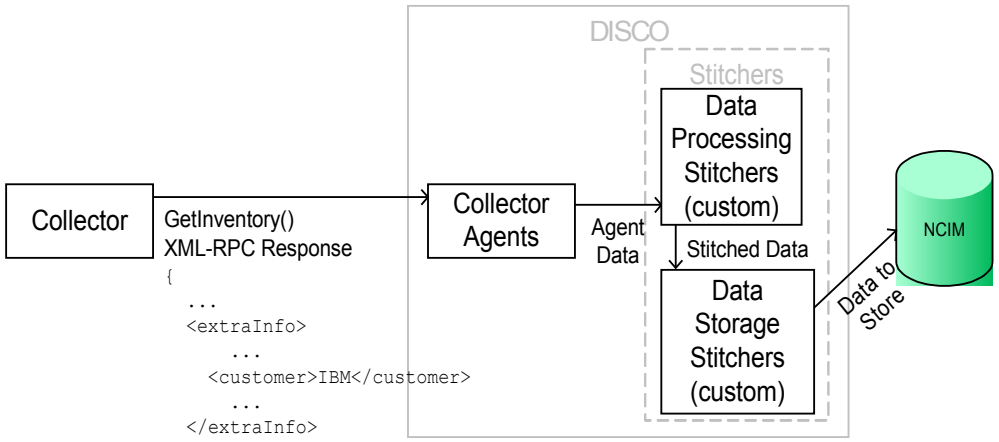


Example of NCIM type data retrieved using freeform 'piggybacking'

Custom data (data lacking DISCO or NCIM support)

Custom data is that which is not natively supported in the XML Schema, DISCO or NCIM. Such data may take the form of extra attributes that you want to store in an extended NCIM database (see NCIM extension via entityDetails in the Network Manager documentation entityDetails) or attributes to be used with custom data processing stitchers (supplied by the collector developer) to resolve any relationship or entity discovery not possible with the standard data.

As with the NCIM data fields, you would typically place custom data in the freeform (extrainfo) sections or in custom XML-RPC call responses. You must also provide custom stitching to process the data.



Category: Custom data
The customer tag above is considered Custom data because it is in the freeform section, and does not match any supported DISCO or NCIM field.

XML Tag -> DISCO Field
No translation of name. The data will appear in DISCO as

```
{
    customer = 'IBM'
}
```

DISCO Field -> NCIM Field
Custom data processing and data storage stitching is performed. Usually resulting in the field being placed in the NCIM entityDetails table.

Writing a Collector

This section explains hows to developi a collector. Consult one of the following language-specific collector library reference documents for more information and working examples:

Java Collector support libraries

Perl Collector support libraries

It is assumed that you already have an XML-RPC server library, that you know what data you need from the EMS in order to satisfy the XML schema (chapter 2), and you know how this data can be retrieved.

It is *not* possible to provide much guidance on EMS-to-Collector data extraction and interpretation or conversion as this is entirely dependent on the EMS in question. You must understand the data present on the EMS and how it can be related to the data required by Network Manager (i.e. how it can be converted into the published XML schema).

There are numerous places on the Internet that can help you create an XML-RPC server from scratch or find a third party library to do the work. Network Manager ships with a Perl XML-RPC server.

Again, remember that there are Java and Perl Collector support libraries to aid Collector development. These libraries are supported by the separate API user-oriented documentation sets listed at the beginning of this section.

Sample collector code

The code presented in this section addresses the following collector responsibilities:

- Implementing, at a minimum, those RPC methods marked as essential, as defined in section 2.1
- Ensuring the data returned by these methods is of the correct format, as defined in section 2.1

The mandatory XML-RPC methods are UpdateData(), GetInfo(), GetDeviceList(), and GetDeviceInfo(). Combined, these methods provide the minimum amount of data that Network Manager requires to represent a device.

Getting the XML data format correct for these methods is a matter of following the XML requirements specified in section 2.

The pseudo code for a collector that supports all the mandatory XML-RPC calls and returns valid data is as follows (*Italics* indicate functionality that is assumed to have been added by the user):

```
// Define our Collector Name, Description
// and supported data sources.
// This Collector only supports a single data source
//
collectorName = "ExampleCollector"
collectorDescr = "An example collector
                  just supporting device list and device info"
dataSource = 1
dataSourceDescr = "the EMS"
```

```
// Setup the XML-RPC Server to support the chosen RPC methods
// and set it into listening mode
//
xmlRpcServerPort = 8080
server = CreateXmlRpcServer(XxmRpcServerPort)
server->RegisterRpcMethod( GetInfo(), &GetInfo() );
server->RegisterRpcMethod( UpdateData(), &UpdateData() );
server->RegisterRpcMethod( GetDeviceList(), &GetDeviceList() );
server->RegisterRpcMethod( GetDeviceInfo(), &GetDeviceInfo() );

server->Listen()   // Never returns
```

The remaining work goes into implementing the RPC methods, as shown below.

```
//
// Implementation of Network Manager supported XML-RPC Methods
//

// UpdateData() Implementation
//
Sub UpdateData( address, subnet )

// Refresh the data
RefreshEMSData( address, subnet );

  Xml = "<updateResponse></updateResponse>"
return xml
End Sub

// GetInfo() Implementation
//
Sub GetInfo()
  Xml = "<collectorInfo>
        <name>" + collectorName + "</name>
        <descr>" + collectorDescr + "</descr>
        <datasources>
          <datasource>
                   <id>" + dataSourceId + "</id>
                   <descr>" + dataSourceDescr + "</descr>
          </datasource>
        </datasources>
       </collectorInfo>"
return xml
End Sub

// GetDeviceList()
//
Sub GetDeviceList( dataSourceId )

  listOfDevices = GetListOfDevicesManagedByEMS( dataSourceId )

Xml = "<deviceList>"

For each entry in listOfDevices
  Xml = xml +
        "<device>
```

```
            <id>" + entry->id + "</id>
            <ip>" + entry->ip + "</ip>
            <addressSpace>" + entry->space + "</addressSpace>
          </device>"
Next entry

  Xml = xml + "</deviceList>"

return xml
End Sub

// GetDeviceInfo
//
Sub GetDeviceInfo( dataSourceId, deviceId )

  device = GetDeviceInfoFromEMS( dataSourceId, deviceId )

  Xml = "<deviceInfo>
           <sysObjectId>device.sysOid</sysObjectId>
           <descr>device.descr</descr>
           <name>device.name</name>
      </deviceInfo>"

return xml
End Sub
```

The XML-RPC Schema reference in sections 2.1 and 2.2 provide further detail on the XML-RPC calls. For example, in GetDeviceList() we return an <id> and <ip>; to find out what a device id is you can look up the GetDeviceList() method in section 2.1 and read the description.

Tip: *The ncp_query_collector tool*

ncp_query_investigator is a Perl based tool that allows you to issue XML-RPC calls to the collector *in development and print the response. This allows a* collector *to be quickly interrogated without the need to run an* Network Manager *discovery, thus speeding up* collector *development.*

The tool works against both Perl and Java based collectors *and has online help accessible by typing 'help' at the prompt.*

The ncp_query_collector tool can be found in the following directory;

$NCHOME/precision/scripts/perl/scripts/

and can be run against the port on which the collector *in development is running as folllows;*

ncp_perl ncp_query_collector.pl -port <collectorPort>

Migration Advice & Change Summary

The Collector framework is backwards compatible: If your Collector works with, say, 3.9 GA then it should work with, say, 4.1.1 FP1 too. As such migration should largely be a matter of copying over your Collector directory to the target system.

You may want to consider updating your custom Collector to take advantage of new features avaiable in the later releases in which case please review the following change summary sections.

DISCO Agent Change Summary

The agents are the main source of DISCO XML-RPC calls to Collectors. Any new XML-RPC calls made by agents will not require mandatory Collector support and so existing Collectors should not be impacted beyond the cost of a call to an unsupported method. Any modified calls will be backwards compatible and so need no change in the Collector.

The main changes (e.g. additional calls) to the Collector Agent since 3.9 GA are listed below to aid users wishing to take advantage of the additional functionality. See the XML-RPC for details of any calls that may have changed.

Agent	Version Introduced	Summary
CollectorLayer1	4.1 GA	New agent: see chapter 3
CollectorRAN	4.1 GA	New agent: see chapter 3
ColectorLTE	4.1.1 GA	New agent: see chapter 3

XML-RPC XML Interface Change Summary

The following changes have been made to the XML-RPC based XML interface (see chapter 2) since the 3.9 GA release.

Note that all changes were backwards compatable, i.e. a Collector that worked in 3.9 GA can be copied over to a more recent release and still function as it did.

The Version Introduced column in the table below shows when the change was introduced: If this is a GA version then the change will also be in later GA releases, and subsequest fix pack releases. For example, a value of '4.1 GA' means the change will be in 4.1 GA, 4.1 FP1 and later fix packs, 4.1.1 GA and later fix packs etc.. If the version is a fix pack (FP) version then the change will be in subsequent fix packs for that major release but NOT other major releases unless otherwise stated.

Method	Version Introduced	Summary
GetInfo	4.1 GA	**New fields:** emsHost, emsName, emsVersion, emsIdentifier, emsRole, emsStatus, extraInfo

GetDeviceInfo	4.1 GA	**New field:** protocol
GetDeviceList	4.1 GA	**New fields:** meId, protocol
GetEntities	3.9 IF1/FP4	**New fields:** ifIndex, interfaceId, cdmType, parent, hwRev, swRev, fwRev
	4.1 GA	as above + serialNumber
GetLayer2VPNs	4.1 GA	**New field:** peerInterfaceId
GetLayer2Connections	4.1 GA	**New fields:** unidirectional, topology
GetLayer1Connections	4.1 GA	**New call**
GetConnections	4.1 GA	**New call**
GetRanData	4.1 GA	**New call**
GetENodeBData	4.1 GA	**New call**
GetMmeData	4.1 GA	**New call**
GetSgwData	4.1 GA	**New call**
GetPgwData	4.1 GA	**New call**
GetHssData	4.1 GA	**New call**
GetPcrfData	4.1 GA	**New call**
GetEirData	4.1 GA	**New call**

Chapter 2: XML Schema & XML-RPC Method Reference

This section details the XML Schema that collectors must adhere to when sending data to Network Manager in response to an XML-RPC call. It describes each of the supported mandatory and optional RPC methods that Network Manager's collector finder and collector agents may issue to the collector.

Mandatory RPC Method Reference

A Collector MUST support the following methods at a minimum if it is to work correctly with Network Manager.

- UpdateData()
- GetInfo()
- GetDeviceList()
- GetDeviceInfo()
- GetAssociatedAddresses()

UpdateData()

The collector implementation of UpdateData() is responsible for ensuring that data is updated as appropriate, i.e. the collector should check the EMS for new data and update any internal stores as appropriate.

If the collector does not need to update its data then it does not need to perform any work other than returning the response message "<updateResponse></updateResponse>".

Note: It is the collector's responsibility to determine which devices the supplied address relates to; the address is NOT guaranteed to be device id of the EMS.

Method Name:	UpdateData
Signature:	(iiss)
Parameters:	(i) Data source id – holds the id of the data source which should be updated. (i) Request Type - determines how the remaining parameters are interpreted. Valid values are; 0 - Update all data (used during an Network Manager Full Discovery) 1 - Update data for single device (specified by Address) 2 - Update data for all devices falling within subnet Address/Mask (s) Address – if mask is not defined then it holds the address of the device to refresh, or else it holds the subnet to refresh. (s) Mask – holds the subnet mask or is not defined. Only

© Copyright IBM Corp. 2014, 2015

	applicable to IPv4 addresses. Example: '255.255.255.0'.
Status:	Mandatory
Usage:	Called by Network Manager's collector Finder prior to requesting data.

Data Definition:
 	
`<?xml version="1.0" encoding="UTF-8"?>`
`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">`
`<xs:element name="updateResponse">`
`<xs:complexType>`
`<xs:sequence>`
`</xs:sequence>`
`</xs:complexType>`
`</xs:element>`
`</xs:schema>`	

Example:
 	
`<updateResponse>`
`</updateResponse>`	

GetInfo()

The collector implementation of GetInfo() is responsible for responding with information on the collector and its supported data sources.

This method is called by the Network Manager's collector finder to determine which collectors are valid and which data sources they support. Network Manager subsequently issues RPC calls for data from the listed data sources.

The collector is responsible for ensuring that each data source has a unique identifier.

Tip: A note on data sources.

The collector framework allows a collector to support multiple data sources, essentially making the collector act as multiple collectors each identified by an identifier unique within that collector. However, a collector is only configured to support a single data source (usually with data source id '1'). This is because in most cases where multiple data sources need to be supported it is more natural to simply run multiple collectors against them.

Method Name:	GetInfo
Signature:	()
Parameters:	N/A
Status:	Mandatory
Usage:	Called by the Network Manager collector finder, once per collector.

Data Definition:
 	
`<?xml version="1.0" encoding="UTF-8"?>`
`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">`
`<xs:element name="collectorInfo">`
`<xs:complexType>`
`<xs:sequence>`
`<!--`
`- name: Name of the Collector`
`- descr: Description of the Collector`
`- datasources: List of data sources known to the Collector.`	

```
      -              Typically there will be only one.
      -->
      <xs:element name="name" type="xs:string" minOccurs="1"/>
      <xs:element name="descr" type="xs:string" minOccurs="0"/>
      <xs:element name="datasources">
        <xs:complexType>
          <xs:sequence>

            <xs:element name="datasource" minOccurs="1" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>

                  <!--
                    - id:     Collectors identifier representing the data source.
                    -         Unique within the Collector.
                    -         Usually just one per Collectors with id '1'.
                    - descr:  Description of the data source represented by <id>
                    -
                    - emsHost: Host of source EMS (IPv4 or DNS name)
                    - emsName: Name of source EMS
                    - emsVersion:    Version of source EMS
                    - emsIdentifier: Custom identifier to represent the EMS
                    - emsRole: Ems role. Valid values are;
                    -                     unknown, primary, backup, other
                    - ensStatus: Ems status. Valid values are;
                    -                     unknown, up, down, other
                    -
                    -->
                  <xs:element name="id" type="xs:string" minOccurs="1"/>
                  <xs:element name="descr" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsHost" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsName" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsVersion" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsIdentifier" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsRole" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsStatus" type="xs:string" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>

          </xs:sequence>
        </xs:complexType>
      </xs:element>

    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<collectorInfo>
     <name>CollectorsName</name>
     <descr>Collectors Description</descr>
     <datasources>
      <datasource>
          <id>1</id>
          <descr>Example data source</descr>
          <emsHost>mybox.ibm.com</emsHost>
          <emsName>AnEMS</emsName>
          <emsVersion>1.0</emsVersion>
          <emsIdentifier>myEmsId</emsIdentifier>
          <emsRole>primary</emsRole>
          <emsStatus>up</emsStatus>
      </datasource>
          .. other data sources ..
     </datasources>
</collectorInfo>
```

GetDeviceList()

The Collector implementation of GetDeviceList() is responsible for determining and returning the addresses and identifiers of all devices that are managed by the Collector within the specified data source.

The device identifier <id> should uniquely identify the device within the collector/data source.

Note that non-IP devices can be supported by putting the non-IP address in the ip field and setting the protocol field to an appropriate value (i.e. 4 to indicate an address supplied by an EMS).

Network Manager supplies the device identifiers returned by GetDeviceList() in further calls to the Collector.

Tip: Device Identifiers

The Device Id field is intended to store the data source's native identifier for the device - i.e. the tag by which the EMS refers to the device in its data. It is possible that where multiple EMSs manage the same device this native identifier differs between collectors.

Method Name:	GetDeviceList
Signature:	(i)
Parameters:	(i) integer - Data source id
Status:	Mandatory
Usage:	Called by the Network Manager' collector finder, once per data source per Collector

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="deviceList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="device" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
              - id:     Identifier that uniquely identifies the device
              -         within the Collector. This is considered the 'native id',
              -         i.e. the identifier used to represent the device in the
              -         Collector (and ideally in the originating EMS).
              - meId:   Optional custom idenifier similar to 'id' but doesn't
              -         need to be the native id used by the Collector/EMS.
              -         This field can be of use when integrating managed devices
              -         (managed elemetns) with external systems
              - ip:     Address of the device.
              -        Note: This doesn't have to be an IP address.
              - protocol:      The protocol to which the address in 'ip' belongs.
              -                Valid values are;
              -                  0 - Unknown
              -                  1 - IPv4
              -                  2 - NAT
              -                  3 - IPv6
              -                  4 - EMS Supplied Address (i.e. the value in
              -                      <ip> is a general string based identifier.
              - addressSpace:  Address space to which the address belongs
            -->
            <xs:element name="id" type="xs:string" minOccurs="1"/>
            <xs:element name="meId" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="ip" type="xs:string" minOccurs="1"/>
            <xs:element name="protocol" type="xs:integer" minOccurs="0"/>
            <xs:element name="addressSpace" type="xs:string" minOccurs="0"/>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<deviceList>
    <device>
        <id>10.1.230.3</id>
        <meId>customIdForManagedDeviceA</meId>
        <ip>10.1.230.3</ip>
        <addressSpace></addressSpace>
    </device>
    .. other devices ..
</deviceList>
```

GetDeviceInfo()

The collector implementation of GetDeviceInfo() is responsible for gathering and returning basic device level information for a specified device and data source.

Name:	GetDeviceInfo
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Identifier
Status:	Mandatory
Usage:	Called by the Network Manager CollectorDetails agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="deviceInfo">
  <xs:complexType>
    <xs:all>

      <!--
        - name:         Name uniquely identifying the device
        - sysObjectId:  Device type identifier used by Network Manager to determine
        -               how to handle the device.
        -               See RFC1213-MIB sysObjectID.
        - ipForwarding: The IP forwarding capability of the device.
        -               See RFC1213-MIB ipForwarding.
        - sysName:      Administratively assigned name.
        -               See RFC1213-MIB sysName.
        - descr:        Description of the device.
        -               See RFC1213-MIB sysDescr.
        - extraInfo:    (optional) Any freeform data including geographic location
      -->
      <xs:element name="name" type="xs:string" minOccurs="1"/>
      <xs:element name="sysObjectId" type="xs:string" minOccurs="1"/>
      <xs:element name="ipForwarding" type="xs:integer" minOccurs="1"/>
      <xs:element name="sysName" type="xs:string" minOccurs="0"/>
      <xs:element name="descr" type="xs:string" minOccurs="0"/>

      <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
        <xs:complexType>
          <xs:sequence>
            <xs:any maxOccurs=”unbounded”>
          </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>

    </xs:all>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<deviceInfo>
    <sysObjectId>1.3.6.1.4.1.0.1.94</sysObjectId>
    <descr>Vendor Device Description (sysDescr)</descr>
    <name>10.1.254.1</name>
</deviceInfo>
```

GetAssociatedAddresses()

The collector implementation of GetAssociatedAddresses() is responsible for gathering and returning all IP addresses configured on a given device id and data source.

This method is used by the CollectorInventory agents to produce a list of associated IP addresses to allow Network Manager to perform address translation.

Name:	GetAssociatedAddresses
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by the Network Manager CollectorInventory agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="associatedAddressList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="addressEntry" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
                - ifIndex:      Index uniquely identifying the interface within
                -               a device. Usually an RFC1213-MIB ifIndex.
                -               MUST be specified if interfaceId is not specified.
                - interfaceId:  A string based unique interface identifier.
                -               MUST be specified if ifIndex is not specified.
                - ifOperStatus: Operational status of the interface.
                -               See RFC1213-MIB ifOperStatus for acceptable values.
                - ipAddress:    IP (V4) address associated with the interface
                -
            -->
            <xs:element name="ifIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="interfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="ifOperStatus" type="xs:integer" minOccurs="1"/>
            <xs:element name="ipAddress" type="xs:string" minOccurs="1"/>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<associatedAddressList>
    <addressEntry>
        <ifOperStatus>1</ifOperStatus>
        <ifIndex>10</ifIndex>
        <ipAddress>10.1.4.1</ipAddress>
    </addressEntry>
    <addressEntry>
        <ifOperStatus>1</ifOperStatus>
        <interfaceId> MyUniqueInterfaceIdFor19</interfaceId>
        <ipAddress>10.1.4.1</ipAddress>
    </addressEntry>
    .. other entries ..
</associatedAddressList>
```

Optional RPC Method Reference

A Collector may optionally support any of the following methods:

- GetInventory()
- GetEntities()
- GetLayer3Vpns()
- GetMplsInterfaces()
- GetLayer3Connections()
- GetLayer2Connections()
- GetLayer1Connections()
- GetConnections()
- GetRanData()
- GetENodeBData()
- GetMmeData()
- GetSgwData()
- GetPgwData()
- GetHssData()
- GetEirData()
- GetPcrfData()

Note: GetDeviceInfo() & GetInventory() vs GetEntities(): Why am I defining interfaces/chassis data in two places?

The XML Schema details two XML-RPC methods that both support chassis data; GetDeviceInfo() and GetEntities(). There are also two methods that support interface data; GetInventory() and GetEntities(). This text attempts to explain the reason behind this apparent redundancy.

Historically an Network Manager discovery discovers devices via SNMP using data from the 'system' and 'ifTable' sections of RFC1213-MIB, the former providing chassis data and the latter providing physical and logical interface data. It also discovers physical entity data using the 'entPhysicalTable' of the

ENTITY-MIB (including (but not limited to) chassis and ports(interfaces)).

GetDeviceInfo() and GetInventory() reflect the RFC1213-MIB discovery for chassis and interfaces respectively. The data from GetDeviceInfo(), which is a mandatory method, provides Network Manager with the early information it needs to coordinate the discovery. The data from GetInventory() represents both physical and logical interfaces and specialises in interface specific attributes such as those used in connectivity processing.

GetEntities() reflects the ENTITY-MIB discovery; it represents a range of physical entities including chassis and ports (interfaces) and is of most importance in entity containment processing. One useful ability that the collector has is to also return logical interface data along with physical interface data in the GetEntities() response, thus allowing logical interfaces to be included in the containment model.

In order to take advantage of Network Manager containment stitching you should ensure the chassis/interface is represented by GetEntities() and to take full advantage of topology stitching you should ensure that interfaces are represented by GetInventory().

GetInventory()

The Collector implementation of GetInventory() is responsible for gathering and returning device interface information for a specified device id and data source.

Name:	GetInventory
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorInventory agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="deviceInventory">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="deviceId" type="xs:string"/>
      <xs:element name="interfaceList">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="interface" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:all>

                  <!--
                    - ifIndex:      See RFC1213 ifIndex. MUST be specified if
                    -               interfaceId is not specified.
                    - interfaceId:  A string based unique interface identifier.
                    -               MUST be specified if ifIndex is not specified.
                    - card:         Card number (with port this should uniquely
                    -               identify the interface)
                    - port:         Port number (with card this should uniquely
                    -               identify the interface)
```

```
                    - ifOperStatus:  See RFC1213 ifOperStatus
                    - ifAdminStatus: See RFC1213 ifAdminStatus
                    - ifPhysAddress: See RFC1213 ifPhysAddress
                    - ifDescr:       See RFC1213 ifDescr
                    - ifName:        See RFC1213 ifName
                    - ifType:        See RFC1213 ifType
                    - isConnected:   See RFC1213 isConnected
                    - ipAddress:     See RFC1213 ipAdEntAddr
                    - netmask:       Number of Host bits in the subnet
                    - subnet:        Subnet address
                    - subnetMask:    Subnet mask
                    - extraInfo:     (optional) Any freeform data
                  -->
                  <xs:element name="ifIndex" type="xs:integer" minOccurs="0"/>
                  <xs:element name="interfaceId" type="xs:string" minOccurs="0"/>
                  <xs:element name="card" type="xs:integer" minOccurs="0"/>
                  <xs:element name="port" type="xs:integer" minOccurs="0"/>
                  <xs:element name="ifOperStatus" type="xs:integer" minOccurs="0"/>
                  <xs:element name="ifAdminStatus" type="xs:integer minOccurs="0""/>
                  <xs:element name="ifPhysAddress" type="xs:integer" minOccurs="0"/>
                  <xs:element name="ifDescr" type="xs:string" minOccurs="0"/>
                  <xs:element name="ifName" type="xs:string" minOccurs="0"/>
                  <xs:element name="ifType" type="xs:integer" minOccurs="0"/>
                  <xs:element name="isConnected" type="xs:integer" minOccurs="0"/>
                  <xs:element name="ipAddress" type="xs:string" minOccurs="0"/>
                  <xs:element name="netmask" type="xs:integer" minOccurs="0"/>
                  <xs:element name="subnet" type="xs:string" minOccurs="0"/>
                  <xs:element name="subnetMask" type="xs:string" minOccurs="0"/>

                  <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:any maxOccurs=”unbounded”>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                </xs:all>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<deviceInventory>
    <deviceId>10.1.254.1</deviceId>
    <interfaceList>
        <interface>
            <ifPhysAddress>00:99:00:E5:C4:06</ifPhysAddress>
            <ifDescr>FastEthernet0/1.10</ifDescr>
            <ifOperStatus>1</ifOperStatus>
            <ifIndex>19</ifIndex>
            <interfaceId>MyUniqueInterfaceIdFor19</interfaceId>
            <ifType>135</ifType>
            <ipAddress></ipAddress>
            <ifName>Fa0/1.10</ifName>
        </interface>
        .. other intefaces ..
    </interfaceList>
</deviceInventory>
```

GetEntities()

The Collector implementation of GetEntities() is responsible for gathering and returning entity information for a given device id and data source. The data

supports both an ENTITY-MIB data form and a more generic non-ENTITY-MIB form for convinience.

This method is used by the CollectorInventory agent to produce a list of entity information for later use in containment modeling.

Name:	GetEntities
Signature:	(is)
Parameters:	integer - Data source id string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorInventory agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="entityData">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="entityList">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entity" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:all>

                  <!--
                    - Fields applicable to both ENTITYT-MIB and Generic Styles
                    - name:  The name of the entity. If ENTITY-MIB style is
                    -        used then it is the equivalent of entPhysicalName.
                    - descr: Entity description. If ENTITY-MIB style is
                    -        used then it is the equivalent of entPhysicalDescr.
                    - alias: Entity alias value. If ENTITY-MIB style is
                    -        used then it is the equivalent of entPhysicalAlias.
                    - parentRelPos: The relative position of the entity within
                    -        the parent. If ENTITY-MIB style is used then
                    -        it is the equivalent of entPhysicalParentRelPos.
                    - fwRev: Entity firmware version. If ENTITY-MIB style used then
                    -        it is the equivalent of entPhysicalFirmwareRev.
                    - hwRev: Entity hardware version. If ENTITY-MIB style used then
                    -        it is the equivalent of entPhysicalHardwareRev.
                    - swRev: Entity software version. If ENTITY-MIB style used then
                    -        it is the equivalent of entPhysicalSoftwareRev.
                    - serialNumber: Entity serial number. If ENTITY-MIB style used
                    -        then it is the equivalent of entPhysicalSerialNum.
                    - extraInfo: (optional) freeform data
                  -->
                  <xs:element name="name" type="xs:string" minOccurs="1"/>
                  <xs:element name="descr" type="xs:string" minOccurs="0"/>
                  <xs:element name="alias" type="xs:string" minOccurs="0"/>
                  <xs:element name="parentRelPos" type="xs:integer" minOccurs="1"/>
                  <xs:element name="fwRev" type="xs:string" minOccurs="0"/>
                  <xs:element name="hwRev" type="xs:string" minOccurs="0"/>
                  <xs:element name="swRev" type="xs:string" minOccurs="0"/>
                  <xs:element name="serialNumber" type="xs:string" minOccurs="0"/>

                   <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:any maxOccurs=”unbounded”>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <!--
                    - ENTITY-MIB Style Specific Fields
                    - index:  See ENTITY-MIB entPhysicalIndex
                    - class:  See ENTITY-MIB entPhysicalClass
```

```
                    - containedIn:  See ENTITY-MIB entPhysicalContainedIn
                    - vendorType:   See ENTITY-MIB entPhysicalVendorType
                  -->
                  <xs:element name="index" type="xs:integer" minOccurs="1"/>
                  <xs:element name="class" type="xs:string" minOccurs="1"/>
                  <xs:element name="containedIn" type="xs:integer" minOccurs="1"/>
                  <xs:element name="vendorType" type="xs:string" minOccurs="0"/>

                  <!--
                    - Generic Style
                    - parent:  The name of the parent entity
                    - cdmType:   The entity type (based on the CDM 'type'
                    - Supported values are;
                    -             0 : Other
                    -             1 : Unknown
                    -             2 : Chassis
                    -             3 : Backplane
                    -             4 : Slot
                    -             5 : Power Supply
                    -             6 : Fan
                    -             7 : Sensor
                    -             8 : Module
                    -             9 : Port
                    -             10 : Rack
                  -->
                  <xs:element name="parent" type="xs:string" minOccurs="1"/>
                  <xs:element name="cdmType" type="xs:integer" minOccurs="1"/>

                </xs:all>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

ENTITY-MIB Style Example:

```
<entityData>
    <entityList>
        <entity>
            <name>ACME 2000 9-slot Chassis System</name>
            <class>3</class>
            <index>1</index>
            <vendorType>1.3.6.1.4.1.1234.12.175</vendorType>
            <descr>AME-2000</descr>
            <parentRelPos>-1</parentRelPos>
        </entity>
        .. other entities ..
    </entityList>
</entityData>
```

Generic Example:

```
<entityData>
    <entityList>
        <entity>
            <name>ACME 2000 9-slot Chassis System/name>
            <cdmType>3</cdmType>
            <parent>0</parent>
            <descr>AME-2000</descr>
            <parentRelPos>0</parentRelPos>
        </entity>
        .. other entities ..
    </entityList>
</entityData>
```

GetLayer3Vpns()

The Collector implementation of GetLayer3Vpns() is responsible for gathering and returning all layer 3 VPNs configured on a given device identifier and data source.

Network Manager processes this data using standard MPLS stitchers to generate VPN membership information for display in the GUI.

Name:	GetLayer3Vpns
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorVpn agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="layer3Vpns">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="vpn" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

           <!--
              - vpnName:  VPN name
              - vrfName:  VRF supporting the VPN
              - rd:       Route distinguisher
              - descr:    VPN description
              - status:   VPN status
              - importRTs:  Imported route targets
              - exportRTs:  Exported route targets
              - interfaces: Associated intefaces (CE facing)
              - extraInfo: (optional) freeform data
            -->
            <xs:element name="vpnName" type="xs:string" minOccurs="0"/>
            <xs:element name="vrfName" type="xs:string" minOccurs="1"/>
            <xs:element name="rd" type="xs:string" minOccurs="1"/>
            <xs:element name="descr" type="xs:string" minOccurs="0"/>
            <xs:element name="status" type="xs:integer" minOccurs="0"/>

            <xs:element name="importRTs" minOccurs="1">
              <xs:complexType>
                <xs:sequence>
                    <xs:element name="rt" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>

            <xs:element name="exportRTs" minOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="rt" type="xs:string"
                              minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>

            <xs:element name="interfaces">
              <xs:complexType>
                <xs:sequence>
                    <xs:element name="interface"
                                minOccurs="0" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
```

```
                          <!--
                            - ifIndex:      ifIndex MUST be specified if
                            -               interfaceId is not specified.
                            - interfaceId:  String based interface identifier.
                            -               MUST be specified if no ifIndex
                            - ceIp:         Address of CE connected to ifIndex
                            - protocol:     Textual description of PE-CE protocol
                            - customerName: Customer name
                            - customerDescr:  Customer description
                          -->
                          <xs:element name="ifIndex"
                                      type="xs:integer" minOccurs="0"/>
                          <xs:element name="interfaceId"
                                      type="xs:string" minOccurs="0"/>
                          <xs:element name="ceIp"
                                      type="xs:string" minOccurs="0"/>
                          <xs:element name="protocol"
                                      type="xs:string" minOccurs="0"/>
                          <xs:element name="customerName"
                                      type="xs:string" minOccurs="0"/>
                          <xs:element name="customerDescr"
                                      type="xs:string" minOccurs="0"/>
                          <xs:element name=”extraInfo”
                                      minOccurs=”0” maxOccurs=”1”>
                            <xs:complexType>
                              <xs:sequence>
                                <xs:any maxOccurs=”unbounded”>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>

                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>

              <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
                <xs:complexType>
                  <xs:sequence>
                    <xs:any maxOccurs=”unbounded”>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>

            </xs:all>
          </xs:complexType>
        </xs:element>

      </xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:schema>
```

Example:

```
<layer3Vpns>
    <vpn>
        <descr>my blue VPN</descr>
        <vrfName>blue</vrfName>
        <vpnName>blue</vpnName>
        <rd>10:401</rd>
        <<status>1</status>
        <importRTs>
            <rt>10:401</rt>
        </importRTs>
        <exportRTs>
            <rt>10:601</rt>
        </exportRTs>
        <interfaces>
            <interface>
                <customerName>A Company</customerName>
                <ceIp>10.0.0.1</ceIp>
```

```
                <ifIndex>10</ifIndex>
                <customerDescr> A Company</customerDescr>
                <protocol>2</protocol>
            </interface>
            <interface>
                <customerName>A Company</customerName>
                <ceIp>10.0.0.1</ceIp>
                <interfaceId>MyeIdFor19</interfaceId>
                <customerDescr> A Company</customerDescr>
                <protocol>2</protocol>
            </interface>
            .. other interfaces ..
        </interfaces>
    </vpn>
    .. other vpns ..
</layer3Vpns>
```

GetLayer2Vpns()

The Collector implementation of GetLayer2Vpns() is responsible for gathering and returning all layer 2 VPNs configured on a given device identifier and data source.

Network Manager processes this data using standard MPLS stitchers to generate VPN membership information for display in the GUI.

Name:	GetLayer2Vpns
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorVpn agent.

Data Definition:

```
<!--
  - GetLayer2VpnsResponse.xsd
  -
  -  This schema represents the XML response to the
  -  GetLayer2Vpns() method.
  -
-->
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="layer2Vpns">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="vpn" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
              - vpnName:      Name of the VPN
              - vcId:         Circuit Id.
              - vcType:       0=invalid, 1= framerelay, 2=atm aal5 vcc,
              -               3=atm trans, 4=vlan, 5=Ethernet, 6=hdlc, 7=ppp,
              -               8=cep, 9=atm vcc, 10=atm vpl, 11 vpls
              -               Common values are 5 (for VPWS) and 11 (for VPLS)
              - peerIp:       IP address of remote device.
              - peerIfIndex:  Remote interface connected to the service
              -               MUST be specified if peerIp is specified
              -               but peerInterfaceId is not specified.
              - peerInterfaceId: String based interface identifier. An alternative
              -               to peerIfIndex for non-IfIndex supporting devices.
              -               MUST be specified if peerIp is specified
              -               but peerIfIndex is not specified.
              - localIfIndex: Local interface connected to the service
              -               MUST be specified if localInterfaceid is not specified.
              - localInterfaceId: String based interface identifier. An alternative
```

```
              -               to localIfIndex for non-IfIndex supporting devices.
              -               MUST be specified if localIfIndex is not specified.
              - status:       0=unknown, 1=down, 2=up
              - topologyType: 0=invalid, 1=other, 2=bgp ip, 3=bgp layer 2,
              -               4=bgp vpls. 5=l2 circuit, 6=ldp vpls, 7=optical,
              -               8=vp ocx, 9=ccc, 10=atm.
              -               Common values are 3 or 5 when vcType is 5 (for VPWS)
              -               and 4 or 6 when vcType is 11 (VPLS).
              - localLabel:   Label imposed for local->remote LSP
              - peerLabel:    Label imposed for remote->local LSP
              - extraInfo:    (optional) Any freeform data
            -->
            <xs:element name="vpnName" type="xs:string" minOccurs="1"/>
            <xs:element name="vcId" type="xs:string" minOccurs="1"/>
            <xs:element name="vcType" type="xs:integer" minOccurs="1"/>
            <xs:element name="peerIp" type="xs:string" minOccurs="0"/>
            <xs:element name="peerIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="peerInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="localIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="localInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="status" type="xs:integer" minOccurs="0"/>
            <xs:element name="topologyType" type="xs:integer" minOccurs="0"/>
            <xs:element name="localLabel" type="xs:integer" minOccurs="1"/>
            <xs:element name="peerLabel" type="xs:integer" minOccurs="1"/>

            <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
              <xs:complexType>
                <xs:sequence>
                  <xs:any maxOccurs=”unbounded”>
                </xs:sequence>
              </xs:complexType>
            </xs:element>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<layer2Vpns>
    <vpn>
        <vpnName>VPWS50</vpnName>
        <localLabel>26</localLabel>
        <vcId>50</vcId>
        <localIfIndex>32</localIfIndex>
        <topologyType>5</topologyType>
        <vcType>5</vcType>
        <peerLabel>28</peerLabel>
        <status>1</status>
        <peerIp>10.1.254.2</peerIp>
        <peerIfIndex>5</peerIfIndex>
    </vpn>
    <vpn>
        <vpnName>VPWS50</vpnName>
        <localLabel>26</localLabel>
        <vcId>50</vcId>
        <localInterfaceId>MyInterfaceId32</localInterfaceI>
        <topologyType>5</topologyType>
        <vcType>5</vcType>
        <peerLabel>28</peerLabel>
        <status>1</status>
        <peerIp>10.1.254.2</peerIp>
        <peerInterfaceId>MyInterfaceId5</peerInterfaceId>
    </vpn>
    .. other vpns ..
</layer2Vpns>
```

GetMplsInterfaces()

The Collector implementation of GetMplsInterfaces() is responsible for gathering and returning the identifiers of all interfaces on a given device id and data source that support MPLS label forwarding.

Network Manager processes this data using standard MPLS stitchers to generate MPLS Core membership information for display in the GUI.

Name:	GetMplsInterfaces
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorVpn agent.

Data Definition:

```
<!--
  - GetMplsInterfaces.xsd
  -
  -  This schema represents the XML response to the
  -  GetMplsInterfaces() method.
  -
-->
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="mplsInterfaceData">
  <xs:complexType>
    <xs:sequence>
    <!--
      - deviceId:     The EMS identifier for the device to which the
      -               MPLS interfaces belong
      -->
      <xs:element name="deviceId" type="xs:string" minOccurs="1"/>
      <xs:element name="interfaceList">
        <xs:complexType>
          <xs:sequence>

            <xs:element name="interface" minOccurs="1" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>

                  <!--
                    - ifIndex:     The RFC1213::IfIndex for the inteface
                    -              MUST be present if interfaceId not present.
                    - interfaceId: A string that uniquely identifiers the interface
                    -              on the device.
                    -              MUST be present if ifIndex not present.
                    -->
                  <xs:element name="ifIndex" type="xs:string" minOccurs="0"/>

                  <xs:element name="interfaceid" type="xs:string" minOccurs="0"/>

                </xs:sequence>
              </xs:complexType>
            </xs:element>

          </xs:sequence>
        </xs:complexType>
      </xs:element>

    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<mplsInterfaceData>
  <deviceId>10.1.1.3</deviceId>
  <interfaceList>
    <interface>
      <ifIndex>15</ifIndex>
    </interface>
    <interface>
      <interfaceId>Gi10/1</interfaceId>
    </interface>
  </interfaceList>
</mplsInterfaceData>
```

GetLayer3Connections()

The Collector implementation of GetLayer3Connections() is responsible for gathering and returning all resolved layer 3 link information for the supplied device id and data source.

Network Manager processes this data resulting in layer 3 connectivity information in the GUI.

Name:	GetLayer3Connections
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorLayer3 agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="layer3Connections">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="connection" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
              - dest:         Destination name/IP
              - destIfIndex:  Destination ifIndex
              - destInterfaceId: String based interface identifier.
              -               MUST be specified if destIfIndex is not specified.
              - src:          Source name/IP
              - srcIfIndex:   Source ifIndex
              - srcInterfaceId: String based interface identifier.
              -               MUST be specified if srcIfIndex is not specified.
              - extraInfo:    (optional) Any freeform data
            -->
            <xs:element name="dest" type="xs:string" minOccurs="1"/>
            <xs:element name="destIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="destInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="src" type="xs:string" minOccurs="0"/>
            <xs:element name="srcIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="srcInterfaceId" type="xs:string" minOccurs="0"/>

            <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
               <xs:complexType>
                    <xs:sequence>
                         <xs:any maxOccurs=”unbounded”>
                    </xs:sequence>
                </xs:complexType>
             </xs:element>
```

```
          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Returned XML:

```
<layer3Connections>
    <connection>
        <dest>10.1.230.1</dest>
        <destIfIndex>22</destIfIndex>
        <srcIfIndex>9</srcIfIndex>
        <src>10.1.254.1</src>
    </connection>
    <connection>
        <dest>10.1.230.1</dest>
        <destInterfaceId>MyUniqueId22</destInterfaceId>
        <srcInterfaceId>MyUniqueId9</srcInterfaceId>
        <src>10.1.254.1</src>
    </connection>
</layer3Connections>
```

GetLayer2Connections()

The Collector implementation of GetLayer2Connections() is responsible for gathering and returning all resolved layer 2 connection information for the supplied device data source.

Network Manager processes this data resulting in layer 2 connectivity information in the GUI.

Name:	GetLayer2Connections
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorLayer2 agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="layer2Connections">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="connection" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
              - dest:         Destination name/IP
              - destIfIndex:  Destination ifIndex
              - destInterfaceId: String based interface identifier.
              -               MUST be specified if destIfIndex is not specified.
              - src:          Source name/IP
              - srcIfIndex:   Source ifIndex
              - srcInterfaceId: String based interface identifier.
              -               MUST be specified if srcIfIndex is not specified.
              - extraInfo:    (optional) Any freeform data
            -->
            <xs:element name="dest" type="xs:string" minOccurs="1"/>
            <xs:element name="destIfIndex" type="xs:integer" minOccurs="0"/>
```

```
            <xs:element name="destInterfaceId" type="xs:string" minOccurs="1"/>
            <xs:element name="src" type="xs:string" minOccurs="0"/>
            <xs:element name="srcIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="srcInterfaceId" type="xs:string" minOccurs="1"/>

             <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
               <xs:complexType>
                    <xs:sequence>
                         <xs:any maxOccurs=”unbounded”>
                    </xs:sequence>
                </xs:complexType>
             </xs:element>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<layer2Connections>
    <connection>
        <dest>10.1.230.1</dest>
        <destIfIndex>22</destIfIndex>
        <srcIfIndex>9</srcIfIndex>
        <src>10.1.254.1</src>
    </connection>
    <connection>
        <dest>10.1.230.1</dest>
        <destInterfaceId>MyUniqueId22</destInterfaceId>
        <srcInterfaceId>MyUniqueId9</srcInterfaceId>
        <src>10.1.254.1</src>
    </connection>
</layer2Connections>
```

GetLayer1Connections()

The Collector implementation of GetLayer1Connections() is responsible for gathering and returning all resolved layer 1 connection information for the supplied device data source.

Network Manager processes this data resulting in layer 1 connectivity information in the GUI.

Name:	GetLayer1Connections
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager’s CollectorLayer1 agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="layer2Connections">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="connection" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
```

```
              - dest:         Destination name or IP.
              - destIfIndex:  Destination ifIndex
              - destInterfaceId: String based interface identifier.
              -
              - src:          Source name or IP
              - srcIfIndex:   Source ifIndex
              - srcInterfaceId: String based interface identifier.
              -
              - unidirectional: 1 (unidirectional) or 0 (bidirectional)
              -
              - extraInfo:    (optional) Any freeform data
            -->
            <xs:element name="dest" type="xs:string" minOccurs="1"/>
            <xs:element name="destIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="destInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="src" type="xs:string" minOccurs="0"/>
            <xs:element name="srcIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="srcInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="unidirectional" type="xs:integer" minOccurs="0"/>
             <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
               <xs:complexType>
                    <xs:sequence>
                         <xs:any maxOccurs=”unbounded”>
                    </xs:sequence>
                </xs:complexType>
             </xs:element>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<layer1Connections>
    <connection>
        <dest>10.1.230.1</dest>
        <destInterfaceId>MyInterface22</destInterfaceId>
        <src>10.1.254.1</src>
        <srcInterfaceId>myInterface9</srcInterfaceId>
        <unidirectional>1</unidirectional>
    </connection>
    <connection>
        .. etc...
    </connection>
</layer1Connections>
```

GetConnections()

The Collector implementation of GetConnections() can be used to retrieve connection information in a standard format for any topology layer.

Note that the root element name is defined as the concatenation of the lower-case topology name and 'Connections'. For example, when called with Topology name 'Microwave', the root element name will be 'microwaveConnections'. Available topology names are dependent upon the collector implementation.

Name:	GetConnections
Signature:	(iss)
Parameters:	(i) integer - Data source id (s) string - Device Id (s) string - Topology name (eg 'Layer3', 'Microwave')
Status:	Optional

Usage:	Called by Network Manager's CollectorLayer1 agent to retrieve Microwave connections, and by the CollectorRAN agent to retrieve logical RAN connections

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="<topologyName>Connections">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="connection" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>

            <!--
              - dest:         Destination name or IP.
              - destIfIndex:  Destination ifIndex
              - destInterfaceId: String based interface identifier.
              -
              - src:          Source name or IP
              - srcIfIndex:   Source ifIndex
              - srcInterfaceId: String based interface identifier.
              -
              - unidirectional: 1 (unidirectional) or 0 (bidirectional)
              -
              - extraInfo:    (optional) Any freeform data
            -->
            <xs:element name="dest" type="xs:string" minOccurs="1"/>
            <xs:element name="destIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="destInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="src" type="xs:string" minOccurs="0"/>
            <xs:element name="srcIfIndex" type="xs:integer" minOccurs="0"/>
            <xs:element name="srcInterfaceId" type="xs:string" minOccurs="0"/>
            <xs:element name="unidirectional" type="xs:integer" minOccurs="0"/>
             <xs:element name=”extraInfo” minOccurs=”0” maxOccurs=”1”>
               <xs:complexType>
                    <xs:sequence>
                         <xs:any maxOccurs=”unbounded”>
                    </xs:sequence>
                </xs:complexType>
             </xs:element>

          </xs:all>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Example:

```
<microwaveConnections>
    <connection>
        <dest>10.1.230.1</dest>
        <destInterfaceId>MyInterface22</destInterfaceId>
        <src>10.1.254.1</src>
        <srcInterfaceId>myInterface9</srcInterfaceId>
        <unidirectional>1</unidirectional>
    </connection>
    <connection>
        <dest>10.1.230.2</dest>
    </connection>
    <connection>
        .. etc...
    </connection>
</microwaveConnections>
```

GetRanData()

The GetRanData() XML-RPC call returns RAN data that is representative of the radio area network (RAN) NCIM model.

Name:	GetRanData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's CollectorRAN agent to gather RAN data for NCIM

Data Definition:

```
In the case of GetRanData() the returned freeform XML data should be representitive
of the RAN NCIM model.

For example, for the ranBaseStation NCIM table which is defined as;

        CREATE TABLE ranBaseStation
        (
            entityId                INTEGER  NOT NULL,
            ranTechnologyType       VARCHAR(10) CONSTRAINT ck_BSRANType
                                        CHECK( ranTechnologyType IN ( 'Unknown',
                                        'Other', 'GSM', 'GPRS' , 'UMTS')),
            baseStationId           VARCHAR(64) NOT NULL,
        )

the XML within the 'calledMethod' tags would be;

        <ranBaseStation>
            <ranTechnologyType>GPRS</ ranTechnologyType>
            <baseStationId>base1234</baseStationId>
        </ranBaseStation>

Note that a value for entityId is not specified as this is automatically generated
when NCIM is populated.
```

Example:

```
<GetRanData>
    <ranBaseStation>
        <ranTechnologyType>GPRS</ ranTechnologyType>
        <baseStationId>base1234</baseStationId>
    </ranBaseStation>
</GetRanData>
```

GetENodeBData()

The Collector implementation of GetENodeBData() is responsible for returning information relating to the Evolved NodeB (eNodeB) network element of a Long Term Evolution (LTE) mobile communications network. A device implementing eNodeB should be identified by returning the lteFunction value "ENODEB" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the eNodeB and its associated cells, sectors and antenna.

Name:	GetENodeBData()
Signature:	(is)
Parameters:	(i) integer - Data source id

	(s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="eNodeBData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the eNodeB (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the eNodeB physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the eNodeB is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from
      the equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
     (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="1"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="1"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of eNodeB functions supported by the physical chassis
    - eNodBId = Unique identifier of the eNodeB
    - eNodBName = User friendly name of the eNodeB
    - maximumOutputPower = The maximum output power of the eNodeB in Watts
    - userCapacity = Maximum number of active users supported by the eNodeB
    - eNodBId = Unique identifier of the eNodeB
    A single eNodeB function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the eNodeB
    - vendorName = The name of the eNodeB vendor
    - operationalState = The operational state of the eNodeB
    - administrativeState = The administrative state of the eNodeB
    - vendorModuleType = Vendor specific eNodeB type
    - softwareVersion = Vendor specific eNodeB software version
    - distinguishedName = The EMS distinguished name of the eNodeB
    - emsIpAddress = The IP address of the element management system
    - backHaulConnection = The IP address of the first hop backhaul device to which
      the enbFunction is connected, for example the IP address of a cell-site router.
    -->
        <xs:element name="enbFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="enbFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="eNodeBId" type="xs:string" minOccurs="1"/>
                    <xs:element name="eNodeBName" type="xs:string" minOccurs="0"/>
                    <xs:element name="maximumOutputPower"
                                type="xs:integer" minOccurs="0"/>
                    <xs:element name="userCapacity" type="xs:integer" minOccurs="0"/>
                    <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="plmn"
```

```
                                        minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                  <xs:sequence>
                                    <xs:element name="mcc"
                                                type="xs:string" minOccurs="0"/>
                                    <xs:element name="mnc"
                                                type="xs:string" minOccurs="0"/>
                                  </xs:sequence>
                                </xs:complexType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="supportedPLMNs" minOccurs="0"/>
                        <xs:element name="vendorName" minOccurs="1">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:enumeration value="Nokia-Siemens"/>
                              <xs:enumeration value="Alcatel-Lucent"/>
                              <xs:enumeration value="Ericsson"/>
                              <xs:enumeration value="Huawei"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="operationalState" minOccurs="0">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:enumeration value="Unknown"/>
                              <xs:enumeration value="Enabled"/>
                              <xs:enumeration value="Disabled"/>
                              <xs:enumeration value="Other"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="administrativeState" minOccurs="0">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:enumeration value="Unknown"/>
                              <xs:enumeration value="Locked"/>
                              <xs:enumeration value="Unlocked"/>
                              <xs:enumeration value="Shutting Down"/>
                              <xs:enumeration value="Other"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="vendorModuleType"
                                    type="xs:string" minOccurs="0"/>
                        <xs:element name="softwareVersion"
                                    type="xs:string" minOccurs="0"/>
                        <xs:element name="distinguishedName"
                                    type="xs:string" minOccurs="0"/>
                        <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
                        <xs:element name="backHaulConnection"
                                    type="xs:string" minOccurs="0"/>
   <!--
    This section lists each logical LTE interface supported by the eNodeB function
    - interfaceType = The type of LTE logical interface
    - lteInterfaceDescription = Description of the LTE logical interface. Should
contain
      the interface type and be unique within the physical chassis.
    - ipAddress = The IP address of the logical interface
    - subNet = The subnetwork of the logical interface
    - vLan = The VLAN to which the logical interface belongs
    -->
                        <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="lteInterface"
                                          minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                  <xs:sequence>
                                    <xs:element name="interfaceType" minOccurs="1">
                                      <xs:simpleType>
                                        <xs:restriction base="xs:string">
```

```
                                    <xs:enumeration value="S1-MME"/>
                                    <xs:enumeration value="S1-U"/>
                                    <xs:enumeration value="X2"/>
                                    <xs:enumeration value="OAM"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                              <xs:element name="lteInterfaceDescription"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="parentInterface"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="ipAddress"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="subNet"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="vLan"
                                          type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
<!--
 This section contains the adjacent eNodeB functions which are connected via an
 X2 interface
 - adjENodeBId = Adjacent eNodeB identifier
 - adjENodeBName = Adjacent eNodeB name
 - adjENodeBIpAddr = Adjacent eNodeB IP address
 -->
                  <xs:element name="x2NeighbourList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="x2Neighbour"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="adjENodeBID"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="adjENodeBName"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="adjENodeBIpAddr"
                                          type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
<!--
 This section contains data on eUtranCells supported by this eNodeB function
 - eUtranCellId = Identifier of the eUtranCell. Often constructed from eNodeB
   eNodeB ID + Physical Cell ID
 - eUtranCellName = The user firendly name of the eUtranCell
 - physicalCellId = The physical cell id of the eUtranCell (integer 0 to 503)
 - localCellId = Locall cell id, an integer value unique within the eNodeB
 - channelBandwidthDl = Downlink channel bandwidth in MHz
 - channelBandwidthUl = Uplink channel bandwidth in MHz
 - maximumOutputPower = The maximum output power of the eUtranCell in Watts
 - userCapacity = Maximum number of active users supported by the eUtranCell
 - earfcnDl = E-UTRA Absolute Radio Frequency Channel Number (downlink). An
   integer value which identifies the downlink carrier frequency of the cell.
 - earfcnUl = E-UTRA Absolute Radio Frequency Channel Number (uplink). An
   integer value which identifies the uplink carrier frequency of the cell.
 - tac = Tracking area code of tracking area which contains the eUtranCell
 - operationalState = The operational state of the eUtranCell
 - administrativeState = The administrative state of the eUtranCell
 - sectorId = The identifier of the sector which supports the eUtranCell
 - sectorName = The name of the sector which supports the eUtranCell
 - distinguishedName = The distinguished name by which the eUtranCell is known
   to its element management system (EMS)
 -->
                  <xs:element name="eUtranCellList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
```

```
                        <xs:sequence>
                          <xs:element name="eUtranCell"
                                      minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="eUtranCellId"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="eUtranCellName"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="physicalCellId"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="localCellId"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="channelBandwidthDl"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="channelBandwidthUl"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="maximumOutputPower"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="userCapacity"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="earfcnDl"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="earfcnUl"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="tac"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="operationalState" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Unknown"/>
                                      <xs:enumeration value="Enabled"/>
                                      <xs:enumeration value="Disabled"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                                <xs:element name="administrativeState" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Unknown"/>
                                      <xs:enumeration value="Locked"/>
                                      <xs:enumeration value="Unlocked"/>
                                      <xs:enumeration value="Shutting Down"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                                <xs:element name="sectorId"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="sectorName"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="distinguishedName"
                                            type="xs:string" minOccurs="0"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
  <!--
   This section contains date on sectors supported by this eNodeB function
   - eUtranCellId = Identifier of the eUtranCell. Often constructed from
                    eNodeB ID + Physical Cell ID
   - sectorId = Sector identifier
   - sectorName = Sector Name
   - sectorNumber = Numeric id of the sector
   - frequencyBand = An integer value which identifies the frequency band supported
                     by the sector. All cells serviced by the sector must have
                     carrier frequencies falling within this band.
   - maximumOutputPower = The maximum available sector power in watts
   - distinguishedName = The distinguished name by which the sector is known to
                         its element management system (EMS)
   - operationalState = The operational state of the sector
```

```
    - administrativeState = The administrative state of the sector
    -->
                    <xs:element name="eUtranSectorList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="eUtranSector"
                                      minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sectorId"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="sectorName"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="sectorNumber"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="frequencyBand"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="maximumOutputPower"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="distinguishedName"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="operationalState" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Unknown"/>
                                      <xs:enumeration value="Enabled"/>
                                      <xs:enumeration value="Disabled"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                                <xs:element name="administrativeState" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Unknown"/>
                                      <xs:enumeration value="Locked"/>
                                      <xs:enumeration value="Unlocked"/>
                                      <xs:enumeration value="Shutting Down"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
    <!--
     Each sector is supported by one or more physical antenna, this section lists the
     identifiers (serial numbers) of the antanna supporting this sector.
     - antennaSerialNumber = Serial number of an antenna supporting this sector
     -->
                                <xs:element name="antennaList"
                                            minOccurs="0" maxOccurs="1">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="antennaSerialNumber"
                                                  minOccurs="0"
                                                  maxOccurs="unbounded"/>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
    <!--
     This section contains information on the antenna functions which support the
     enbFunction
     - antennaSerialNumber = Unique antenna identifier
     - antennaHeight = This is the vertical height of the antenna in metres above
                       WGS84 datum surface
     - antennaDownTilt = The antenna vertical tilt in degrees.
     - antennaBearing = The bearing in degrees that the antenna is pointing in.
     - antennaMaxAzimuth = The maximum amount of change of azimuth the system
       system can support. This is the change in degrees clockwise from bearing.
     - antennaMinAzimuth = The minimum amount of change of azimuth the system
```

```
      can support. This is the change in degrees clockwise from bearing.
    - antennaHorizontalBeamwidth = The power beamwidth of the antenna pattern in
      the horizontal plane. A value of 360 indicates an omni-directional antenna.
      A single integral value corresponding to an angle in degrees between 0 and 360.
    - antennaVerticalBeamwidth = The power beamwidth of the antenna pattern in
      the vertical plane. A value of 360 indicates an omni-directional antenna.
      A single integral value corresponding to an angle in degrees between 0 and 360.
    - antennaLatitude = Latitude of Antenna equipment. This is the angular distance
      (east and west) from the prime meridian on the earth's surface. e.g. 35.832636
      Based on WGS84 standard.
    - antennaLongitude = Longitude of Antenna equipment. This is the angular
      distance (north and south) from the equator on the earth's surface
      e.g. -78.838753. Based on WGS84 standard.
    - antennaLocationName = The antenna location name
    - antennaManufacturer = The vendor/manufacturer of the antenna
    - antennaModel = Vendor specific antenna model type
    - distinguishedName = The distinguished name by which the anenna is known to
      its element management system (EMS)
    -->
                    <xs:element name="antennaFunctionList"
                                minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="antennaFunction"
                                      minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="antennaSerialNumber"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="antennaHeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaDownTilt"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaBearing"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaMaxAzimuth"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaMinAzimuth"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaHorizontalBeamwidth"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaVerticalBeamwidth"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="antennaLatitude"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="antennaLongitude"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="antennaLocationName"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="antennaManufacturer"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="antennaModel" type="xs:string"
                                            minOccurs="0"/>
                                <xs:element name="distinguishedName"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="sectorIdList"
                                            minOccurs="0" maxOccurs="1">
                  <!--
                   Each antennFunction supports one or more sectors, this section lists the
                   sectorIds of those supported sectors
                   - sectorId = Supported sector identifier
                   -->
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="sectorId"
                                                  minOccurs="0"
                                                  maxOccurs="unbounded"/>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
```

```
                    </xs:complexType>
                  </xs:element>
<!--
 This section contains information on the S1-MME interfaces which connect this
 enbFunction to EPC mmeFunctions.
 - mmeName = The name of the mmeFunction connected over this S1-MME
   interface
 - mmeCode= The code which niquely identifies the MME within its MME group
 - mmeGroupId = Uniquely identifies the MME Group within the PLMN
 - mmeMcc = Mobile country code of the connected MME
 - mmeMnc = Mobile network code of the connected MME
 - mmeIpAddress = The IP address on the connected MME which supports
   this S1-MME interface
 - mmeLinkWeight= An indication of the relative probability that the link will be
   used from the perspective of the enbFunction. The weight is expressed
   as a percentage.
 - mmeLinkUsage = Indicates the usage of the S1-MME link, where available
   and applicable
 -->
                  <xs:element name="connectedMMEList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="connectedMME"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mmeName"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="mmeCode"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="mmeGroupId"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="mmeMcc"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="mmeMnc"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="mmeIpAddress"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="mmeLinkWeight"
                                          type="xs:integer" minOccurs="0"/>
                              <xs:element name="mmeLinkUsage" minOccurs="0">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="Primary"/>
                                    <xs:enumeration value="Secondary"/>
                                    <xs:enumeration value="Backup"/>
                                    <xs:enumeration value="Other"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
<!--
 This section contains information on the S1-U interfaces which connect this
 enbFunction to EPC sgwFunctions.
 - sgwName = The name of the sgwFunction connected over this S1-U
   interface
 - sgwIpAddress = The IP address on the connected SGW which supports
   this S1-U interface
 - sgwLinkWeight= An indication of the relative probability that the link will be
   used from the perspective of the enbFunction. The weight is expressed
   as a percentage.
 - sgwLinkUsage = Indicates the usage of the S1-U link, where available
   and applicable
 -->
                  <xs:element name="connectedSGWList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="connectedSGW"
                                    minOccurs="0" maxOccurs="unbounded">
```

```
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sgwName"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="sgwIpAddr"
                                            type="xs:string" minOccurs="0"/>
                                <xs:element name="sgwLinkWeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="sgwLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
  <!--
   This section contains information on the OAM interfaces which connect this
   enbFunction to an EMS
   - oamName = The name of the EMS connected over this OAM interface
   - oamIpAddress = The IP address on the connected EMS which supports
     this OAM interface
   - oamLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the enbFunction. The weight is expressed
     as a percentage.
   - oamLinkUsage = Indicates the usage of the OAM link, where available
     and applicable
   -->
                    <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="oamConnection"
                                      minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="oamName"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamIpAddr"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamLinkWeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
</xs:schema>
```

Example:

```
<<eNodeBData>
  <parentChassisId>ENB Chassis 1</parentChassisId>
  <parentChassisDn>PLMN/CHASSIS-1</parentChassisDn>
  <locationName>Southbank Tower</locationName>
  <latitude>10.0000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>6</altitude>
  <timezoneOffset>105</timezoneOffset>
  <enbFunctionList>
    <enbFunction>
      <eNodeBId>10001</eNodeBId>
      <eNodeBName>ENodeB-1</eNodeBName>
      <maximumOutputPower>150</maximumOutputPower>
      <userCapacity>500</userCapacity>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>ENODEB</vendorModuleType>
      <softwareVersion>V1.0</softwareVersion>
      <distinguishedName>PLMN/CHASSIS-1/ENB-10001</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <backHaulConnection>12.34.56.90</backHaulConnection>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S1-MME</interfaceType>
          <lteInterfaceDescription>S1-MME-1</lteInterfaceDescription>
          <parentInterface>ETH-1-2</parentInterface>
          <ipAddress>12.34.56.78</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>601</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S1-U</interfaceType>
          <lteInterfaceDescription>S1-U</lteInterfaceDescription>
          <parentInterface>ETH-1-2</parentInterface>
          <ipAddress>12.34.56.78</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>601</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>X2</interfaceType>
          <lteInterfaceDescription>X2</lteInterfaceDescription>
          <parentInterface>ETH-1-2</parentInterface>
          <ipAddress>12.34.56.78</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>111</vLan>
        </lteInterface>
      </lteInterfaceList>
      <x2NeighbourList>
        <x2Neighbour>
          <adjENodeBID>10002</adjENodeBID>
          <adjENodeBName>ENodeB-2</adjENodeBName>
          <adjENodeBIpAddr>12.34.56.91</adjENodeBIpAddr>
        </x2Neighbour>
        <x2Neighbour>
          <adjENodeBID>10003</adjENodeBID>
          <adjENodeBName>ENodeB-3</adjENodeBName>
          <adjENodeBIpAddr>12.34.56.92</adjENodeBIpAddr>
        </x2Neighbour>
      </x2NeighbourList>
      <eUtranCellList>
        <eUtranCell>
```

```
          <eUtranCellId>1000111</eUtranCellId>
          <eUtranCellName>Cell-10001-1-1</eUtranCellName>
          <physicalCellId>1</physicalCellId>
          <localCellId>1</localCellId>
          <channelBandwidthDl>20</channelBandwidthDl>
          <channelBandwidthUl>20</channelBandwidthUl>
          <maximumOutputPower>20</maximumOutputPower>
          <userCapacity>50</userCapacity>
          <earfcnDl>1111</earfcnDl>
          <earfcnUl>2222</earfcnUl>
          <tac>3333</tac>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <sectorId>100011</sectorId>
          <sectorName>Sector-10001-1</sectorName>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/CELL-1</distinguishedName>
        </eUtranCell>
        <eUtranCell>
          <eUtranCellId>1000122</eUtranCellId>
          <eUtranCellName>Cell-10001-2-2</eUtranCellName>
          <physicalCellId>2</physicalCellId>
          <localCellId>2</localCellId>
          <channelBandwidthDl>20</channelBandwidthDl>
          <channelBandwidthUl>20</channelBandwidthUl>
          <maximumOutputPower>20</maximumOutputPower>
          <userCapacity>50</userCapacity>
          <earfcnDl>1111</earfcnDl>
          <earfcnUl>2222</earfcnUl>
          <tac>3333</tac>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <sectorId>100012</sectorId>
          <sectorName>Sector-10001-2</sectorName>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/CELL-2</distinguishedName>
        </eUtranCell>
        <eUtranCell>
          <eUtranCellId>1000133</eUtranCellId>
          <eUtranCellName>Cell-10001-3-3</eUtranCellName>
          <physicalCellId>3</physicalCellId>
          <localCellId>3</localCellId>
          <channelBandwidthDl>20</channelBandwidthDl>
          <channelBandwidthUl>20</channelBandwidthUl>
          <maximumOutputPower>20</maximumOutputPower>
          <userCapacity>50</userCapacity>
          <earfcnDl>1111</earfcnDl>
          <earfcnUl>2222</earfcnUl>
          <tac>3333</tac>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <sectorId>100013</sectorId>
          <sectorName>Sector-10001-3</sectorName>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/CELL-3</distinguishedName>
        </eUtranCell>
      </eUtranCellList>
      <eUtranSectorList>
        <eUtranSector>
          <sectorId>100011</sectorId>
          <sectorName>Sector-10001-1</sectorName>
          <sectorNumber>1</sectorNumber>
          <frequencyBand>5</frequencyBand>
          <maximumOutputPower>100</maximumOutputPower>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/SECTOR-1</distinguishedName>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <antennaList>
            <antennaSerialNumber>10101</antennaSerialNumber>
          </antennaList>
        </eUtranSector>
        <eUtranSector>
          <sectorId>100012</sectorId>
          <sectorName>Sector-10001-2</sectorName>
          <sectorNumber>2</sectorNumber>
          <frequencyBand>5</frequencyBand>
          <maximumOutputPower>100</maximumOutputPower>
```

```
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/SECTOR-2</distinguishedName>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <antennaList>
            <antennaSerialNumber>10102</antennaSerialNumber>
          </antennaList>
        </eUtranSector>
        <eUtranSector>
          <sectorId>100013</sectorId>
          <sectorName>Sector-10001-3</sectorName>
          <sectorNumber>3</sectorNumber>
          <frequencyBand>5</frequencyBand>
          <maximumOutputPower>100</maximumOutputPower>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/SECTOR-3</distinguishedName>
          <operationalState>Enabled</operationalState>
          <administrativeState>Unlocked</administrativeState>
          <antennaList>
            <antennaSerialNumber>10103</antennaSerialNumber>
          </antennaList>
        </eUtranSector>
      </eUtranSectorList>
      <antennaFunctionList>
        <antennaFunction>
          <antennaSerialNumber>10101</antennaSerialNumber>
          <antennaHeight>10</antennaHeight>
          <antennaDownTilt>10</antennaDownTilt>
          <antennaBearing>270</antennaBearing>
          <antennaMaxAzimuth>30</antennaMaxAzimuth>
          <antennaMinAzimuth>5</antennaMinAzimuth>
          <antennaHorizontalBeamwidth>30</antennaHorizontalBeamwidth>
          <antennaVerticalBeamwidth>30</antennaVerticalBeamwidth>
          <antennaLatitude>10.000</antennaLatitude>
          <antennaLongitude>-20.000</antennaLongitude>
          <antennaLocationName>Southbank Tower</antennaLocationName>
          <antennaManufacturer>AERIAL</antennaManufacturer>
          <antennaModel>1234</antennaModel>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/ANT-1</distinguishedName>
          <sectorIdList>
            <sectorId>100011</sectorId>
          </sectorIdList>
        </antennaFunction>
         <antennaFunction>
          <antennaSerialNumber>10102</antennaSerialNumber>
          <antennaHeight>10</antennaHeight>
          <antennaDownTilt>10</antennaDownTilt>
          <antennaBearing>150</antennaBearing>
          <antennaMaxAzimuth>30</antennaMaxAzimuth>
          <antennaMinAzimuth>5</antennaMinAzimuth>
          <antennaHorizontalBeamwidth>30</antennaHorizontalBeamwidth>
          <antennaVerticalBeamwidth>30</antennaVerticalBeamwidth>
          <antennaLatitude>10.000</antennaLatitude>
          <antennaLongitude>-20.000</antennaLongitude>
          <antennaLocationName>Southbank Tower</antennaLocationName>
          <antennaManufacturer>AERIAL</antennaManufacturer>
          <antennaModel>1234</antennaModel>
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/ANT-2</distinguishedName>
          <sectorIdList>
            <sectorId>100012</sectorId>
          </sectorIdList>
        </antennaFunction>
         <antennaFunction>
          <antennaSerialNumber>10103</antennaSerialNumber>
          <antennaHeight>10</antennaHeight>
          <antennaDownTilt>10</antennaDownTilt>
          <antennaBearing>30</antennaBearing>
          <antennaMaxAzimuth>30</antennaMaxAzimuth>
          <antennaMinAzimuth>5</antennaMinAzimuth>
          <antennaHorizontalBeamwidth>30</antennaHorizontalBeamwidth>
          <antennaVerticalBeamwidth>30</antennaVerticalBeamwidth>
          <antennaLatitude>10.000</antennaLatitude>
          <antennaLongitude>-20.000</antennaLongitude>
          <antennaLocationName>Southbank Tower</antennaLocationName>
          <antennaManufacturer>AERIAL</antennaManufacturer>
          <antennaModel>1234</antennaModel>
```

```
          <distinguishedName>PLMN/CHASSIS-1/ENB-10001/ANT-3</distinguishedName>
          <sectorIdList>
            <sectorId>100013</sectorId>
          </sectorIdList>
        </antennaFunction>
      </antennaFunctionList>
      <connectedMMEList>
        <connectedMME>
          <mmeName>MME-1</mmeName>
          <mmeCode>1</mmeCode>
          <mmeGroupId>6000</mmeGroupId>
          <mmeMcc>123</mmeMcc>
          <mmeMnc>456</mmeMnc>
          <mmeIpAddress>12.34.56.78</mmeIpAddress>
          <mmeLinkWeight>100</mmeLinkWeight>
          <mmeLinkUsage>Primary</mmeLinkUsage>
        </connectedMME>
        <connectedMME>
          <mmeName>MME-2</mmeName>
          <mmeCode>2</mmeCode>
          <mmeGroupId>6000</mmeGroupId>
          <mmeMcc>123</mmeMcc>
          <mmeMnc>456</mmeMnc>
          <mmeIpAddress>12.34.56.89</mmeIpAddress>
          <mmeLinkWeight>0</mmeLinkWeight>
          <mmeLinkUsage>Secondary</mmeLinkUsage>
        </connectedMME>
      </connectedMMEList>
      <oamConnectionList>
        <oamConnection>
          <oamName>EMS-1</oamName>
          <oamIpAddr>23.45.66.77</oamIpAddr>
          <oamLinkWeight>100</oamLinkWeight>
          <oamLinkUsage>Primary</oamLinkUsage>
        </oamConnection>
      </oamConnectionList>
    </enbFunction>
  </enbFunctionList>
</eNodeBData>
```

GetMmeData()

The Collector implementation of GetMmeData() is responsible for returning information relating to the Mobility Management Entity (MME) network element of a Long Term Evolution (LTE) mobile communications network. A device implementing MME functionality should be identified by returning the lteFunction value "MME" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the MME.

Name:	GetMmeData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="mmeData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the
mmeFunction (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the MME physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the MME is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from
the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from
the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
     (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of MME functions supported by the physical chassis
    - mmeName = Unique identifier of the mmeFunction within this physical chassis
    - mmcCode = MME Code: Uniquely identifies an MME within an MME Group
    - mmeGroupId = MME Group Identifier: Uniquely identifies an MME Group
    - mmePoolName = The name of the MME Pool to which this mmeFunction belongs
    A single MME function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the MME
    - vendorName = The name of the MME vendor
    - operationalState = The operational state of the MME
```

```
    - administrativeState = The administrative state of the MME
    - vendorModuleType = Vendor specific MME type
    - softwareVersion = Vendor specific MME software version
    - distinguishedName = The EMS distinguished name of the MME
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="mmeFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="mmeFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="mmeName" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="mmeCode" type="xs:string" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="mmeGroupId" type="xs:string" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="mmePoolName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
                    <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="plmn" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="mcc" type="xs:string"
minOccurs="0"/>
                                <xs:element name="mnc" type="xs:string"
minOccurs="0"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="supportedPLMNs" minOccurs="0"/>
                    <xs:element name="vendorName" minOccurs="1">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Nokia-Siemens"/>
                          <xs:enumeration value="Alcatel-Lucent"/>
                          <xs:enumeration value="Ericsson"/>
                          <xs:enumeration value="Huawei"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="operationalState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Enabled"/>
                          <xs:enumeration value="Disabled"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="administrativeState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Locked"/>
                          <xs:enumeration value="Unlocked"/>
                          <xs:enumeration value="Shutting Down"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="vendorModuleType" type="xs:string"
minOccurs="0"/>
                    <xs:element name="softwareVersion" type="xs:string"
minOccurs="0"/>
                    <xs:element name="distinguishedName" type="xs:string"
```

```
minOccurs="0"/>
                    <xs:element name="emsIpAddress" type="xs:string"
minOccurs="0"/>
   <!--
    This section lists each logical LTE interface supported by the MME function
    - interfaceType = The type of LTE logical interface
    - lteInterfaceDescription = Description of the LTE logical interface. Should
contain
      the interface type and be unique within the physical chassis.
    - ipAddress = The IP address of the logical interface
    - subNet = The subnetwork of the logical interface
    - vLan = The VLAN to which the logical interface belongs
    -->
                    <xs:element name="lteInterfaceList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="lteInterface" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="interfaceType" minOccurs="1"
maxOccurs="1">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="S1-MME"/>
                                      <xs:enumeration value="S3"/>
                                      <xs:enumeration value="S6a"/>
                                      <xs:enumeration value="S10"/>
                                      <xs:enumeration value="S11"/>
                                      <xs:enumeration value="S13"/>
                                      <xs:enumeration value="OAM"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                                <xs:element name="lteInterfaceDescription"
type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="parentInterface"
type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="ipAddress" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="subNet" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="vLan" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S1-MME interfaces which connect this
    mmeFunction to EUTRAN enbFunctions.
    - eNodeBName = The name of the enbFunction connected over this interface
    - eNodeBIpAddr = The IP address on the connected enbFunction which supports
this interface
    - s1mmeLinkWeight= An indication of the relative probability that the link
will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s1mmeLinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="s1mmeConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s1mmeConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="eNodeBName" type="xs:string"
minOccurs="0" maxOccurs="1"/>
```

```
                                <xs:element name="eNodeBIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s1mmeLinkWeight"
type="xs:integer" minOccurs="0"/>
                                <xs:element name="s1mmeLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S3 interfaces which connect this
    mmeFunction to an SGSN
    - sgsnName = The name of the SGSN connected over this interface
    - sgsnIpAddr = The IP address on the connected SGSN which supports this
interface
    - s3LinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s3LinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="s3ConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s3Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sgsnName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="sgsnIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s3LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s3LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S6a interfaces which connect this
    mmeFunction to an HSS
    - hssName = The name of the HSS connected over this interface
    - hssIpAddr = The IP address on the connected HSS which supports this
interface
    - s6aLinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s6aLinkUsage = Indicates the usage of the link, where available and
```

```
applicable
    -->
                    <xs:element name="s6aConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s6aConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="hssName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="hssIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s6aLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s6aLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S10 interfaces which connect this
    mmeFunction to a peer mmeFunction
    - mmeName = The name of the peer mmeFunction connected over this interface
    - mmeCode = The MME Code of the peer mmeFunction connected over this interface
    - mmeGroupId = The MME Group Id of the peer mmeFunction connected over this
interface
    - mmeMcc = The MCC of the peer mmeFunction connected over this interface
    - mmeMnc = The MNC of the peer mmeFunction connected over this interface
    - mmeIpAddr = The IP address on the connected mmeFunction which supports this
interface
    - s10LinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s10LinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="s10ConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s10Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="mmeName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="mmeCode" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="mmeGroupId" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="mmeMcc" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="mmeMnc" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="mmeIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s10LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s10LinkUsage" minOccurs="0">
                                  <xs:simpleType>
```

```
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S11 interfaces which connect this
    mmeFunction to an sgwFunction
    - sgwName = The name of the sgwFunction connected over this interface
    - sgwIpAddr = The IP address on the connected sgwFunction which supports this
interface
    - s11LinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s11LinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="s11ConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s11Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sgwName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="sgwIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s11LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s11LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S13 interfaces which connect this
    mmeFunction to an EIR
    - eirName = The name of the EIR connected over this interface
    - eirIpAddr = The IP address on the connected EIR which supports this
interface
    - s13LinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s13LinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="s13ConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
```

```
                          <xs:element name="s13Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="eirName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
                                <xs:element name="eirIpAddr" type="xs:string"
minOccurs="0" maxOccurs="1"/>
                                <xs:element name="s13LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s13LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the OAM interfaces which connect this
    mmeFunction to an EMS
    - oamName = The name of the EMS connected over this interface
    - oamIpAddr = The IP address on the connected EMS which supports this
interface
    - oamLinkWeight= An indication of the relative probability that the link will
be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - oamLinkUsage = Indicates the usage of the link, where available and
applicable
    -->
                    <xs:element name="oamConnectionList" minOccurs="0"
maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="oamConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="oamName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamIpAddr" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
```

```
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<mmeData>
  <parentChassisId>MME-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/MME-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <mmeFunctionList>
    <mmeFunction>
      <mmeName>MME01</mmeName>
      <mmeCode>1</mmeCode>
      <mmeGroupId>22222</mmeGroupId>
      <mmePoolName>MMEPOOL-01</mmePoolName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>MMEMODEL1</vendorModuleType>
      <softwareVersion>MMESW1</softwareVersion>
      <distinguishedName>PLMN/MME-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S1-MME</interfaceType>
          <lteInterfaceDescription>S1-MME</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.67.89</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S3</interfaceType>
          <lteInterfaceDescription>S3</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.67.88</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S6a</interfaceType>
          <lteInterfaceDescription>S6a</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.67.87</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S10</interfaceType>
          <lteInterfaceDescription>S10</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.67.86</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S11</interfaceType>
          <lteInterfaceDescription>S11</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
```

```
          <ipAddress>23.45.67.85</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S13</interfaceType>
          <lteInterfaceDescription>S13</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.67.84</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-3</parentInterface>
          <ipAddress>23.45.67.83</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <s1mmeConnectionList>
        <s1mmeConnection>
          <eNodeBName>ENodeB-01</eNodeBName>
          <eNodeBIpAddr>99.88.77.66</eNodeBIpAddr>
        </s1mmeConnection>
        <s1mmeConnection>
          <eNodeBName>ENodeB-02</eNodeBName>
          <eNodeBIpAddr>99.88.77.55</eNodeBIpAddr>
        </s1mmeConnection>
      </s1mmeConnectionList>
      <s3ConnectionList>
        <s3Connection>
          <sgsnName>SGSN-01</sgsnName>
          <sgsnIpAddr>44.33.22.11</sgsnIpAddr>
        </s3Connection>
      </s3ConnectionList>
      <s6aConnectionList>
        <s6aConnection>
          <hssName>HSS-01</hssName>
          <hssIpAddr>55.33.22.11</hssIpAddr>
        </s6aConnection>
      </s6aConnectionList>
      <s10ConnectionList>
        <s10Connection>
          <mmeName>MME-02</mmeName>
          <mmeCode>6</mmeCode>
          <mmeGroupId>33333</mmeGroupId>
          <mmeMcc>123</mmeMcc>
          <mmeMnc>456</mmeMnc>
          <mmeIpAddr>66.33.22.11</mmeIpAddr>
        </s10Connection>
      </s10ConnectionList>
      <s11ConnectionList>
        <s11Connection>
          <sgwName>SGW-01</sgwName>
          <sgwIpAddr>77.33.22.11</sgwIpAddr>
        </s11Connection>
      </s11ConnectionList>
      <s13ConnectionList>
        <s13Connection>
          <eirName>EIR-01</eirName>
          <eirIpAddr>88.33.22.11</eirIpAddr>
        </s13Connection>
      </s13ConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.22.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </mmeFunction>
  </mmeFunctionList>
</mmeData>
```

GetSgwData()

The Collector implementation of GetSgwData() is responsible for returning information relating to the Serving Gateway (SGW) network element of a Long Term Evolution (LTE) mobile communications network. A device implementing SGW functionality should be identified by returning the lteFunction value "SGW" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the SGW.

Name:	GetSgwData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sgwData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the
      sgwFunction (as returned by GetDeviceList()
    - parentChassisDn = The distinguished name of the SGW physicalChassis
      as known by the EMS
    - locationName = Identifier for the geographic location at which the SGW is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
     (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in
      format UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of SGW functions supported by the physical chassis
    - sgwName = Unique identifier of the sgwFunction within this physical chassis
    A single SGW function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the SGW
    - vendorName = The name of the SGW vendor
```

```
    - operationalState = The operational state of the SGW
    - administrativeState = The administrative state of the SGW
    - vendorModuleType = Vendor specific SGW type
    - softwareVersion = Vendor specific SGW software version
    - distinguishedName = The EMS distinguished name of the SGW
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="sgwFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="sgwFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="sgwName" type="xs:string" minOccurs="1"/>
                    <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="plmn" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                                <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="supportedPLMNs" minOccurs="0"/>
                    <xs:element name="vendorName" minOccurs="1">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Nokia-Siemens"/>
                          <xs:enumeration value="Alcatel-Lucent"/>
                          <xs:enumeration value="Ericsson"/>
                          <xs:enumeration value="Huawei"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="operationalState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Enabled"/>
                          <xs:enumeration value="Disabled"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="administrativeState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Locked"/>
                          <xs:enumeration value="Unlocked"/>
                          <xs:enumeration value="Shutting Down"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="vendorModuleType" type="xs:string" minOccurs="0"/>
                    <xs:element name="softwareVersion" type="xs:string" minOccurs="0"/>
                    <xs:element name="distinguishedName" type="xs:string" minOccurs="0"/>
                    <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
    <!--
     This section lists each logical LTE interface supported by the SGW function
     - interfaceType = The type of LTE logical interface
     - lteInterfaceDescription = Description of the LTE logical interface. Should contain
       the interface type and be unique within the physical chassis.
     - ipAddress = The IP address of the logical interface
     - subNet = The subnetwork of the logical interface
     - vLan = The VLAN to which the logical interface belongs
     -->
                    <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
```

```
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="lteInterface" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="interfaceType" minOccurs="1">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="S1-U"/>
                                      <xs:enumeration value="S11"/>
                                      <xs:enumeration value="S4"/>
                                      <xs:enumeration value="S5"/>
                                      <xs:enumeration value="S8"/>
                                      <xs:enumeration value="OAM"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                                <xs:element name="lteInterfaceDescription"
type="xs:string" minOccurs="1"/>
                                <xs:element name="parentInterface" type="xs:string"
minOccurs="1"/>
                                <xs:element name="ipAddress" type="xs:string"
minOccurs="0"/>
                                <xs:element name="subNet" type="xs:string"
minOccurs="0"/>
                                <xs:element name="vLan" type="xs:string" minOccurs="0"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S1-U interfaces which connect this
    sgwFunction to EUTRAN enbFunctions.
    - eNodeBName = The name of the enbFunction connected over this interface
    - eNodeBIpAddr = The IP address on the connected enbFunction which supports this
interface
    - s1uLinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s1uLinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="s1uConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s1uConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="eNodeBName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="eNodeBIpAddr" type="xs:string"
minOccurs="0"/>
                                <xs:element name="s1uLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s1uLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
```

```
    This section contains information on the S11 interfaces which connect this
    sgwFunction to a peer mmeFunction
    - mmeName = The name of the peer mmeFunction connected over this interface
    - mmeCode = The MME Code of the peer mmeFunction connected over this interface
    - mmeGroupId = The MME Group Id of the peer mmeFunction connected over this interface
    - mmeMcc = The MCC of the peer mmeFunction connected over this interface
    - mmeMnc = The MNC of the peer mmeFunction connected over this interface
    - mmeIpAddr = The IP address on the connected mmeFunction which supports this
interface
    - s11LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s11LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="s11ConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s11Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="mmeName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="mmeCode" type="xs:string"
minOccurs="0"/>
                                <xs:element name="mmeGroupId" type="xs:string"
minOccurs="0"/>
                                <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                                <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                                <xs:element name="mmeIpAddr" type="xs:string"
minOccurs="0"/>
                                <xs:element name="s11LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s11LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S4 interfaces which connect this
    sgwFunction to an SGSN
    - sgsnName = The name of the SGSN connected over this interface
    - sgsnIpAddr = The IP address on the connected SGSN which supports this interface
    - s4LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s4LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="s4ConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s4Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sgsnName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="sgsnIpAddr" type="xs:string"
minOccurs="0"/>
                                <xs:element name="s4LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s4LinkUsage" minOccurs="0">
                                  <xs:simpleType>
```

```
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S5 interfaces which connect this
    sgwFunction to a PGW
    - pgwName = The name of the PGW connected over this interface
    - pgwIpAddr = The IP address on the connected PGW which supports this interface
    - s5LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s5LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="s5ConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s5Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="pgwName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="pgwIpAddr" type="xs:string"
minOccurs="0"/>
                                <xs:element name="s5LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s5LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the S8 interfaces which connect this
    sgwFunction to a PGW
    - pgwName = The name of the PGW connected over this interface
    - pgwIpAddr = The IP address on the connected PGW which supports this interface
    - s8LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s8LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="s8ConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="s8Connection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="pgwName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="pgwIpAddr" type="xs:string"
minOccurs="0"/>
```

```
                                <xs:element name="s8LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s8LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the OAM interfaces which connect this
    sgwFunction to an EMS
    - oamName = The name of the EMS connected over this interface
    - oamIpAddr = The IP address on the connected EMS which supports this interface
    - oamLinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the sgwFunction. The weight is expressed
      as a percentage.
    - oamLinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="oamConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="oamName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamIpAddr" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<sgwData>
  <parentChassisId>SGW-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/SGW-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <sgwFunctionList>
    <sgwFunction>
      <sgwName>SGW01</sgwName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>SGWMODEL1</vendorModuleType>
      <softwareVersion>SGWSW1</softwareVersion>
      <distinguishedName>PLMN/SGW-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S1-U</interfaceType>
          <lteInterfaceDescription>S1-MME</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.66.89</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S11</interfaceType>
          <lteInterfaceDescription>S11</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.66.88</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S5</interfaceType>
          <lteInterfaceDescription>S5</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.66.87</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S8</interfaceType>
          <lteInterfaceDescription>S8</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.66.86</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.66.85</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <s1uConnectionList>
        <s1uConnection>
          <eNodeBName>ENodeB-01</eNodeBName>
          <eNodeBIpAddr>99.88.77.66</eNodeBIpAddr>
```

```
        </s1uConnection>
        <s1uConnection>
          <eNodeBName>ENodeB-02</eNodeBName>
          <eNodeBIpAddr>99.88.77.55</eNodeBIpAddr>
        </s1uConnection>
      </s1uConnectionList>
      <s11ConnectionList>
        <s11Connection>
          <mmeName>MME-01</mmeName>
          <mmeIpAddr>77.33.22.11</mmeIpAddr>
        </s11Connection>
      </s11ConnectionList>
      <s4ConnectionList>
        <s4Connection>
          <sgsnName>SGSN-01</sgsnName>
          <sgsnIpAddr>44.33.22.11</sgsnIpAddr>
        </s4Connection>
      </s4ConnectionList>
      <s5ConnectionList>
        <s5Connection>
          <pgwName>PGW-01</pgwName>
          <pgwIpAddr>55.33.22.11</pgwIpAddr>
        </s5Connection>
      </s5ConnectionList>
      <s8ConnectionList>
        <s8Connection>
          <pgwName>PGW-66</pgwName>
          <pgwIpAddr>77.33.22.11</pgwIpAddr>
        </s8Connection>
      </s8ConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.22.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </sgwFunction>
  </sgwFunctionList>
</sgwData>
```

GetPgwData()

The Collector implementation of GetPgwData() is responsible for returning information relating to the Packet Gateway (aka Packet Data Network Gateway) (PGW) network element of a Long Term Evolution (LTE) mobile communications network. A device implementing PGW functionality should be identified by returning the lteFunction value "PGW" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the PGW.

Name:	GetPgwData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="pgwData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the pgwFunction (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the PGW physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the PGW is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
      (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of PGW functions supported by the physical chassis
    - pgwName = Unique identifier of the pgwFunction within this physical chassis
    A single PGW function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the PGW
    - vendorName = The name of the PGW vendor
    - operationalState = The operational state of the PGW
    - administrativeState = The administrative state of the PGW
    - vendorModuleType = Vendor specific PGW type
    - softwareVersion = Vendor specific PGW software version
    - distinguishedName = The EMS distinguished name of the PGW
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="pgwFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
```

```
            <xs:sequence>
              <xs:element name="pgwFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="pgwName" type="xs:string" minOccurs="1"/>
                    <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="plmn" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                                <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="supportedPLMNs" minOccurs="0"/>
                    <xs:element name="vendorName" minOccurs="1">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Nokia-Siemens"/>
                          <xs:enumeration value="Alcatel-Lucent"/>
                          <xs:enumeration value="Ericsson"/>
                          <xs:enumeration value="Huawei"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="operationalState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Enabled"/>
                          <xs:enumeration value="Disabled"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="administrativeState" minOccurs="0">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="Unknown"/>
                          <xs:enumeration value="Locked"/>
                          <xs:enumeration value="Unlocked"/>
                          <xs:enumeration value="Shutting Down"/>
                          <xs:enumeration value="Other"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="vendorModuleType" type="xs:string" minOccurs="0"/>
                    <xs:element name="softwareVersion" type="xs:string" minOccurs="0"/>
                    <xs:element name="distinguishedName" type="xs:string" minOccurs="0"/>
                    <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
   <!--
    This section lists each logical LTE interface supported by the PGW function
    - interfaceType = The type of LTE logical interface
    - lteInterfaceDescription = Description of the LTE logical interface. Should contain
      the interface type and be unique within the physical chassis.
    - ipAddress = The IP address of the logical interface
    - subNet = The subnetwork of the logical interface
    - vLan = The VLAN to which the logical interface belongs
    -->
                    <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="lteInterface" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="interfaceType" minOccurs="1">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
```

```
                                        <xs:enumeration value="S5"/>
                                        <xs:enumeration value="S8"/>
                                        <xs:enumeration value="Gx"/>
                                        <xs:enumeration value="OAM"/>
                                      </xs:restriction>
                                    </xs:simpleType>
                                  </xs:element>
                                  <xs:element name="lteInterfaceDescription"
type="xs:string" minOccurs="1"/>
                                  <xs:element name="parentInterface" type="xs:string"
minOccurs="1"/>
                                  <xs:element name="ipAddress" type="xs:string"
minOccurs="0"/>
                                  <xs:element name="subNet" type="xs:string" minOccurs="0"/>
                                  <xs:element name="vLan" type="xs:string" minOccurs="0"/>
                                </xs:sequence>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
   <!--
    This section contains information on the S5 interfaces which connect this
    pgwFunction to an SGW
    - sgwName = The name of the SGW connected over this interface
    - sgwIpAddr = The IP address on the connected SGW which supports this interface
    - s5LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s5LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                      <xs:element name="s5ConnectionList" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="s5Connection" minOccurs="0"
maxOccurs="unbounded">
                              <xs:complexType>
                                <xs:sequence>
                                  <xs:element name="sgwName" type="xs:string"
minOccurs="1"/>
                                  <xs:element name="sgwIpAddr" type="xs:string"
minOccurs="1"/>
                                  <xs:element name="s5LinkWeight" type="xs:integer"
minOccurs="0"/>
                                  <xs:element name="s5LinkUsage" minOccurs="0">
                                    <xs:simpleType>
                                      <xs:restriction base="xs:string">
                                        <xs:enumeration value="Primary"/>
                                        <xs:enumeration value="Secondary"/>
                                        <xs:enumeration value="Backup"/>
                                        <xs:enumeration value="Other"/>
                                      </xs:restriction>
                                    </xs:simpleType>
                                  </xs:element>
                                </xs:sequence>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
   <!--
    This section contains information on the S8 interfaces which connect this
    pgwFunction to an SGW
    - sgwName = The name of the SGW connected over this interface
    - sgwIpAddr = The IP address on the connected SGW which supports this interface
    - s8LinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - s8LinkUsage = Indicates the usage of the link, where available and applicable
    -->
                      <xs:element name="s8ConnectionList" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="s8Connection" minOccurs="0"
```

```
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="sgwName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="sgwIpAddr" type="xs:string"
minOccurs="1"/>
                                <xs:element name="s8LinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="s8LinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the Gx interfaces which connect this
    pgwFunction to a PCRF
    - pcrfName = The name of the SGW connected over this interface
    - pcrfIpAddr = The IP address on the connected SGW which supports this interface
    - GxLinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the mmeFunction. The weight is expressed
      as a percentage.
    - GxLinkUsage = Indicates the usage of the link, where available and applicable
    -->
                    <xs:element name="GxConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="GxConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="pcrfName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="pcrfIpAddr" type="xs:string"
minOccurs="1"/>
                                <xs:element name="GxLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="GxLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
   <!--
    This section contains information on the OAM interfaces which connect this
    pgwFunction to an EMS
    - oamName = The name of the EMS connected over this interface
    - oamIpAddr = The IP address on the connected EMS which supports this interface
    - oamLinkWeight= An indication of the relative probability that the link will be
      used from the perspective of the pgwFunction. The weight is expressed
      as a percentage.
    - oamLinkUsage = Indicates the usage of the link, where available and applicable
    -->
```

```
                    <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="oamConnection" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="oamName" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamIpAddr" type="xs:string"
minOccurs="1"/>
                                <xs:element name="oamLinkWeight" type="xs:integer"
minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<pgwData>
  <parentChassisId>PGW-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/PGW-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <pgwFunctionList>
    <pgwFunction>
      <pgwName>SGW01</pgwName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>PGWMODEL1</vendorModuleType>
      <softwareVersion>PGWSW1</softwareVersion>
      <distinguishedName>PLMN/PGW-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S5</interfaceType>
          <lteInterfaceDescription>S5</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.65.89</ipAddress>
```

```
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>S8</interfaceType>
          <lteInterfaceDescription>S8</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.65.88</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>Gx</interfaceType>
          <lteInterfaceDescription>Gx</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.65.87</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.66.85</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <s5ConnectionList>
        <s5Connection>
          <sgwName>SGW-01</sgwName>
          <sgwIpAddr>77.33.22.11</sgwIpAddr>
        </s5Connection>
      </s5ConnectionList>
      <s8ConnectionList>
        <s8Connection>
          <sgwName>SGW-66</sgwName>
          <sgwIpAddr>77.33.21.11</sgwIpAddr>
        </s8Connection>
      </s8ConnectionList>
      <GxConnectionList>
        <GxConnection>
          <pcrfName>PCRF-01</pcrfName>
          <pcrfIpAddr>88.33.20.11</pcrfIpAddr>
        </GxConnection>
      </GxConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.22.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </pgwFunction>
  </pgwFunctionList>
</pgwData>
```

GetHssData()

The Collector implementation of GetHssData() is responsible for returning information relating to the Home Subscriber Server (HSS) network element of a mobile communications network. A device implementing HSS functionality should be identified by returning the lteFunction value "HSS" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the HSS.

Name:	GetHssData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="hssData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the hssFunction (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the HSS physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the HSS is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
      (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of HSS functions supported by the physical chassis
    - hssName = Unique identifier of the hssFunction within this physical chassis
    A single HSS function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the HSS
    - vendorName = The name of the HSS vendor
    - operationalState = The operational state of the HSS
    - administrativeState = The administrative state of the HSS
    - vendorModuleType = Vendor specific HSS type
    - softwareVersion = Vendor specific HSS software version
    - distinguishedName = The EMS distinguished name of the HSS
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="hssFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
```

```
            <xs:element name="hssFunction" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="hssName" type="xs:string" minOccurs="1"/>
                  <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="plmn" minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                              <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="supportedPLMNs" minOccurs="0"/>
                  <xs:element name="vendorName" minOccurs="1">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Nokia-Siemens"/>
                        <xs:enumeration value="Alcatel-Lucent"/>
                        <xs:enumeration value="Ericsson"/>
                        <xs:enumeration value="Huawei"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="operationalState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Enabled"/>
                        <xs:enumeration value="Disabled"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="administrativeState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Locked"/>
                        <xs:enumeration value="Unlocked"/>
                        <xs:enumeration value="Shutting Down"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="vendorModuleType" type="xs:string" minOccurs="0"/>
                  <xs:element name="softwareVersion" type="xs:string" minOccurs="0"/>
                  <xs:element name="distinguishedName" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
<!--
 This section lists each logical LTE interface supported by the HSS function
 - interfaceType = The type of LTE logical interface
 - lteInterfaceDescription = Description of the LTE logical interface. Should contain
   the interface type and be unique within the physical chassis.
 - ipAddress = The IP address of the logical interface
 - subNet = The subnetwork of the logical interface
 - vLan = The VLAN to which the logical interface belongs
 -->
                  <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="lteInterface"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="interfaceType" minOccurs="1">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="S6a"/>
```

```
                                    <xs:enumeration value="OAM"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                              <xs:element name="lteInterfaceDescription"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="parentInterface"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="ipAddress"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="subNet" type="xs:string" minOccurs="0"/>
                              <xs:element name="vLan" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the S6a interfaces which connect this
   hssFunction to an MME
   - mmeName = The name of the MME connected over this interface
   - mmeIpAddr = The IP address on the connected MME which supports this interface
   - s6aLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the mmeFunction. The weight is expressed
     as a percentage.
   - s6aLinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="s6aConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="s6aConnection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mmeName" type="xs:string"
                                          minOccurs="1"/>
                              <xs:element name="mmeIpAddr"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="s6aLinkWeight"
                                          type="xs:integer" minOccurs="0"/>
                              <xs:element name="s6aLinkUsage" minOccurs="0">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="Primary"/>
                                    <xs:enumeration value="Secondary"/>
                                    <xs:enumeration value="Backup"/>
                                    <xs:enumeration value="Other"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the OAM interfaces which connect this
   hssFunction to an EMS
   - oamName = The name of the EMS connected over this interface
   - oamIpAddr = The IP address on the connected EMS which supports this interface
   - oamLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the hssFunction. The weight is expressed
     as a percentage.
   - oamLinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="oamConnection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
```

```
                                <xs:element name="oamName" type="xs:string" minOccurs="1"/>
                                <xs:element name="oamIpAddr"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamLinkWeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<hssData>
  <parentChassisId>HSS-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/HSS-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <hssFunctionList>
    <hssFunction>
      <hssName>SGW01</hssName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>HSSMODEL1</vendorModuleType>
      <softwareVersion>HSSSW1</softwareVersion>
      <distinguishedName>PLMN/HSS-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S6a</interfaceType>
          <lteInterfaceDescription>S6a</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.64.89</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
```

```
          <ipAddress>23.45.64.85</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <s6aConnectionList>
        <s6aConnection>
          <mmeName>MME-01</mmeName>
          <mmeIpAddr>77.33.26.11</mmeIpAddr>
        </s6aConnection>
      </s6aConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.21.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </hssFunction>
  </hssFunctionList>
</hssData>
```

GetPcrfData()

The Collector implementation of GetPcrfData() is responsible for returning information relating to the Policy and Charging Rules Function (PCRF) network element of a mobile communications network. A device implementing PCRF functionality should be identified by returning the lteFunction value "PCRF" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the PCRF.

Name:	GetPcrfData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="pcrfData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the pcrfFunction (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the PCRF physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the PCRF is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
      (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of PCRF functions supported by the physical chassis
    - pcrfName = Unique identifier of the pcrfFunction within this physical chassis
    A single PCRF function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the PCRF
    - vendorName = The name of the PCRF vendor
    - operationalState = The operational state of the PCRF
    - administrativeState = The administrative state of the PCRF
    - vendorModuleType = Vendor specific PCRF type
    - softwareVersion = Vendor specific PCRF software version
    - distinguishedName = The EMS distinguished name of the PCRF
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="pcrfFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
```

```
            <xs:element name="pcrfFunction" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="pcrfName" type="xs:string" minOccurs="1"/>
                  <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="plmn" minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                              <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="supportedPLMNs" minOccurs="0"/>
                  <xs:element name="vendorName" minOccurs="1">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Nokia-Siemens"/>
                        <xs:enumeration value="Alcatel-Lucent"/>
                        <xs:enumeration value="Ericsson"/>
                        <xs:enumeration value="Huawei"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="operationalState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Enabled"/>
                        <xs:enumeration value="Disabled"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="administrativeState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Locked"/>
                        <xs:enumeration value="Unlocked"/>
                        <xs:enumeration value="Shutting Down"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="vendorModuleType" type="xs:string" minOccurs="0"/>
                  <xs:element name="softwareVersion" type="xs:string" minOccurs="0"/>
                  <xs:element name="distinguishedName" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
<!--
 This section lists each logical LTE interface supported by the PCRF function
 - interfaceType = The type of LTE logical interface
 - lteInterfaceDescription = Description of the LTE logical interface. Should contain
   the interface type and be unique within the physical chassis.
 - ipAddress = The IP address of the logical interface
 - subNet = The subnetwork of the logical interface
 - vLan = The VLAN to which the logical interface belongs
 -->
                  <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="lteInterface"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="interfaceType" minOccurs="1">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="Gx"/>
```

```
                                    <xs:enumeration value="OAM"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                              <xs:element name="lteInterfaceDescription"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="parentInterface"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="ipAddress"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="subNet" type="xs:string" minOccurs="0"/>
                              <xs:element name="vLan" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the Gx interfaces which connect this
   pcrfFunction to a PGW
   - pgwName = The name of the PGW connected over this interface
   - pgwIpAddr = The IP address on the connected PGW which supports this interface
   - GxLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the pcrfFunction. The weight is expressed
     as a percentage.
   - GxLinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="GxConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="GxConnection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="pgwName" type="xs:string" minOccurs="1"/>
                              <xs:element name="pgwIpAddr"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="GxLinkWeight"
                                          type="xs:integer" minOccurs="0"/>
                              <xs:element name="GxLinkUsage" minOccurs="0">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="Primary"/>
                                    <xs:enumeration value="Secondary"/>
                                    <xs:enumeration value="Backup"/>
                                    <xs:enumeration value="Other"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the OAM interfaces which connect this
   pcrfFunction to an EMS
   - oamName = The name of the EMS connected over this interface
   - oamIpAddr = The IP address on the connected EMS which supports this interface
   - oamLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the pcrfFunction. The weight is expressed
     as a percentage.
   - oamLinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="oamConnection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="oamName" type="xs:string" minOccurs="1"/>
```

```
                                <xs:element name="oamIpAddr"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamLinkWeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<pcrfData>
  <parentChassisId>PCRF-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/PCRF-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <pcrfFunctionList>
    <pcrfFunction>
      <pcrfName>PCRF01</pcrfName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>PCRFMODEL1</vendorModuleType>
      <softwareVersion>PCRFSW1</softwareVersion>
      <distinguishedName>PLMN/PCRF-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>Gx</interfaceType>
          <lteInterfaceDescription>S6a</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.64.79</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
          <ipAddress>23.45.64.75</ipAddress>
```

```
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <GxConnectionList>
        <GxConnection>
          <pgwName>PGW-01</pgwName>
          <pgwIpAddr>77.33.26.11</pgwIpAddr>
        </GxConnection>
      </GxConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.20.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </pcrfFunction>
  </pcrfFunctionList>
</pcrfData>
```

GetEirData()

The Collector implementation of GetEirData() is responsible for returning information relating to the Equipment Identity Register (EIR) network element of a mobile communications network. A device implementing EIR functionality should be identified by returning the lteFunction value "EIR" as part of the extraInfo returned in the GetDeviceInfo() method invoked for that device.

Network Manager processes the returned data to populate configuration parameters and connectivity information for the EIR.

Name:	GetEirData()
Signature:	(is)
Parameters:	(i) integer - Data source id (s) string - Device Id
Status:	Optional
Usage:	Called by Network Manager's LTE Collector agent.

Data Definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="eirData">
    <xs:complexType>
      <xs:sequence>
  <!--
    - parentChassisId = Identifier of the physicalChassis supporting the eirFunction (as
      by GetDeviceList()
    - parentChassisDn = The distinguished name of the EIR physicalChassis as
      known by the EMS
    - locationName = Identifier for the geographic location at which the EIR is
      located
    - latitude = Angular distance in decimal degrees east (+) and west (-) from the
      prime meridian on the earth's surface. e.g. 35.832636.(WGS84 standard)
    - longitude = Angular distance in decimal degrees north (+) and south (-) from the
      equator on the earth's surface. e.g. -78.838753 (WGS84 standard)
    - altitude = This is the vertical height in metres above WGS84 datum surface
      (EGM96) of the geographical location
    - timezoneOffset = Offset of geographic location local time from UTC in format
      UTC-HH:MM or UTC+HH:MM e.g. UTC+10:30
  -->
        <xs:element name="parentChassisId" type="xs:string" minOccurs="0"/>
        <xs:element name="parentChassisDn" type="xs:string" minOccurs="0"/>
        <xs:element name="locationName" type="xs:string" minOccurs="0"/>
        <xs:element name="latitude" type="xs:string" minOccurs="0"/>
        <xs:element name="longitude" type="xs:string" minOccurs="0"/>
        <xs:element name="altitude" type="xs:string" minOccurs="0"/>
        <xs:element name="timezoneOffset" type="xs:string" minOccurs="0"/>
  <!--
    The list of EIR functions supported by the physical chassis
    - eirName = Unique identifier of the eirFunction within this physical chassis
    A single EIR function can support multiple PLMNs. Each supported PLMN
    is listed here:
        - mcc = The Mobile Country Code (MCC) of the PLMN consisting of
          three digits.
        - mnc = The Mobile Network Code (MNC) of the PLMN consisting of two
          or three digits.
    - supportedPLMNs = The number of PLMNs supported by the EIR
    - vendorName = The name of the EIR vendor
    - operationalState = The operational state of the EIR
    - administrativeState = The administrative state of the EIR
    - vendorModuleType = Vendor specific EIR type
    - softwareVersion = Vendor specific EIR software version
    - distinguishedName = The EMS distinguished name of the EIR
    - emsIpAddress = The IP address of the element management system
    -->
        <xs:element name="eirFunctionList" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
```

```
            <xs:element name="eirFunction" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="eirName" type="xs:string" minOccurs="1"/>
                  <xs:element name="plmnList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="plmn" minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mcc" type="xs:string" minOccurs="0"/>
                              <xs:element name="mnc" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="supportedPLMNs" minOccurs="0"/>
                  <xs:element name="vendorName" minOccurs="1">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Nokia-Siemens"/>
                        <xs:enumeration value="Alcatel-Lucent"/>
                        <xs:enumeration value="Ericsson"/>
                        <xs:enumeration value="Huawei"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="operationalState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Enabled"/>
                        <xs:enumeration value="Disabled"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="administrativeState" minOccurs="0">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Unknown"/>
                        <xs:enumeration value="Locked"/>
                        <xs:enumeration value="Unlocked"/>
                        <xs:enumeration value="Shutting Down"/>
                        <xs:enumeration value="Other"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="vendorModuleType" type="xs:string" minOccurs="0"/>
                  <xs:element name="softwareVersion" type="xs:string" minOccurs="0"/>
                  <xs:element name="distinguishedName" type="xs:string" minOccurs="0"/>
                  <xs:element name="emsIpAddress" type="xs:string" minOccurs="0"/>
<!--
 This section lists each logical LTE interface supported by the EIR function
 - interfaceType = The type of LTE logical interface
 - lteInterfaceDescription = Description of the LTE logical interface. Should contain
   the interface type and be unique within the physical chassis.
 - ipAddress = The IP address of the logical interface
 - subNet = The subnetwork of the logical interface
 - vLan = The VLAN to which the logical interface belongs
 -->
                  <xs:element name="lteInterfaceList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="lteInterface"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="interfaceType" minOccurs="1">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="S13"/>
```

```
                                    <xs:enumeration value="OAM"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                              <xs:element name="lteInterfaceDescription"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="parentInterface" type="xs:string"
                                          minOccurs="1"/>
                              <xs:element name="ipAddress"
                                          type="xs:string" minOccurs="0"/>
                              <xs:element name="subNet" type="xs:string" minOccurs="0"/>
                              <xs:element name="vLan" type="xs:string" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the S13 interfaces which connect this
   eirFunction to an MME
   - mmeName = The name of the MME connected over this interface
   - mmeIpAddr = The IP address on the connected MME which supports this interface
   - s13LinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the eirFunction. The weight is expressed
     as a percentage.
   - s13LinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="s13ConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="s13Connection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="mmeName"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="mmeIpAddr"
                                          type="xs:string" minOccurs="1"/>
                              <xs:element name="s13LinkWeight"
                                          type="xs:integer" minOccurs="0"/>
                              <xs:element name="s13LinkUsage" minOccurs="0">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="Primary"/>
                                    <xs:enumeration value="Secondary"/>
                                    <xs:enumeration value="Backup"/>
                                    <xs:enumeration value="Other"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
  <!--
   This section contains information on the OAM interfaces which connect this
   eirFunction to an EMS
   - oamName = The name of the EMS connected over this interface
   - oamIpAddr = The IP address on the connected EMS which supports this interface
   - oamLinkWeight= An indication of the relative probability that the link will be
     used from the perspective of the eirFunction. The weight is expressed
     as a percentage.
   - oamLinkUsage = Indicates the usage of the link, where available and applicable
   -->
                  <xs:element name="oamConnectionList" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="oamConnection"
                                    minOccurs="0" maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:sequence>
```

```
                                <xs:element name="oamName"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamIpAddr"
                                            type="xs:string" minOccurs="1"/>
                                <xs:element name="oamLinkWeight"
                                            type="xs:integer" minOccurs="0"/>
                                <xs:element name="oamLinkUsage" minOccurs="0">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                      <xs:enumeration value="Primary"/>
                                      <xs:enumeration value="Secondary"/>
                                      <xs:enumeration value="Backup"/>
                                      <xs:enumeration value="Other"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example:

```
<eirData>
  <parentChassisId>EIR-CHASSIS-01</parentChassisId>
  <parentChassisDn>PLMN/EIR-CHASSIS-01</parentChassisDn>
  <locationName>Southbank Switch Site</locationName>
  <latitude>10.00000</latitude>
  <longitude>-20.0000</longitude>
  <altitude>2</altitude>
  <timezoneOffset>102</timezoneOffset>
  <eirFunctionList>
    <eirFunction>
      <eirName>EIR01</eirName>
      <plmnList>
        <plmn>
          <mcc>123</mcc>
          <mnc>456</mnc>
        </plmn>
      </plmnList>
      <supportedPLMNs>1</supportedPLMNs>
      <vendorName>Nokia-Siemens</vendorName>
      <operationalState>Enabled</operationalState>
      <administrativeState>Unlocked</administrativeState>
      <vendorModuleType>EIRMODEL1</vendorModuleType>
      <softwareVersion>EIRSW1</softwareVersion>
      <distinguishedName>PLMN/EIR-01</distinguishedName>
      <emsIpAddress>12.34.56.78</emsIpAddress>
      <lteInterfaceList>
        <lteInterface>
          <interfaceType>S13</interfaceType>
          <lteInterfaceDescription>S13</lteInterfaceDescription>
          <parentInterface>ETHERNET-1</parentInterface>
          <ipAddress>23.45.63.79</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
        <lteInterface>
          <interfaceType>OAM</interfaceType>
          <lteInterfaceDescription>OAM</lteInterfaceDescription>
          <parentInterface>ETHERNET-2</parentInterface>
```

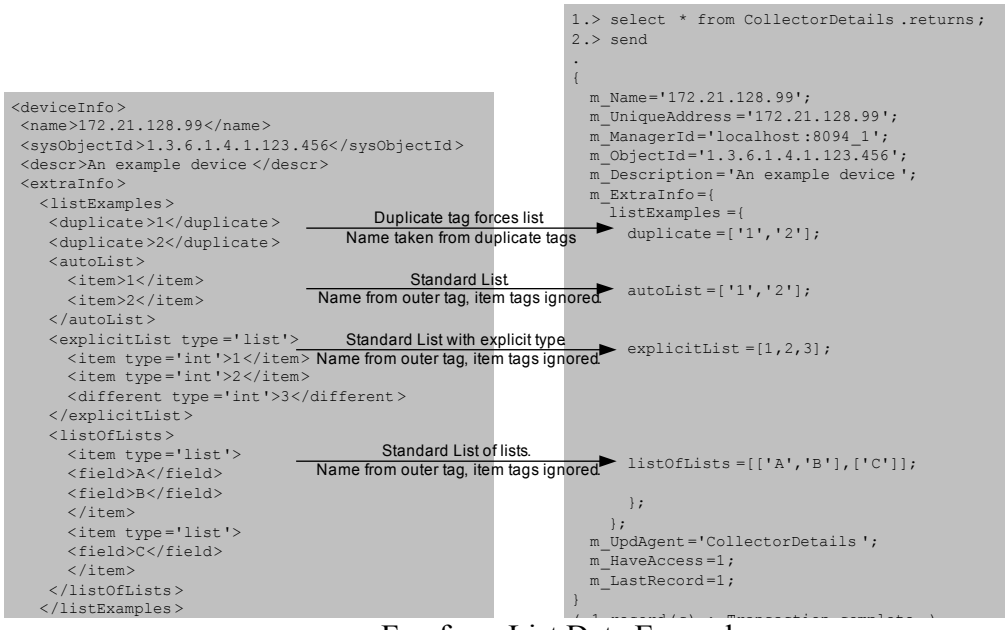
```
          <ipAddress>23.45.63.75</ipAddress>
          <subNet>255.255.255.252</subNet>
          <vLan>2</vLan>
        </lteInterface>
      </lteInterfaceList>
      <s13ConnectionList>
        <s13Connection>
          <mmeName>MME-01</mmeName>
          <mmeIpAddr>77.33.26.11</mmeIpAddr>
        </s13Connection>
      </s13ConnectionList>
      <oamConnectionList>
        <oamConnection>
          <oamName>OAM-01</oamName>
          <oamIpAddr>99.33.20.11</oamIpAddr>
        </oamConnection>
      </oamConnectionList>
    </eirFunction>
  </eirFunctionList>
</eirData>
```

Extending The XML Schema

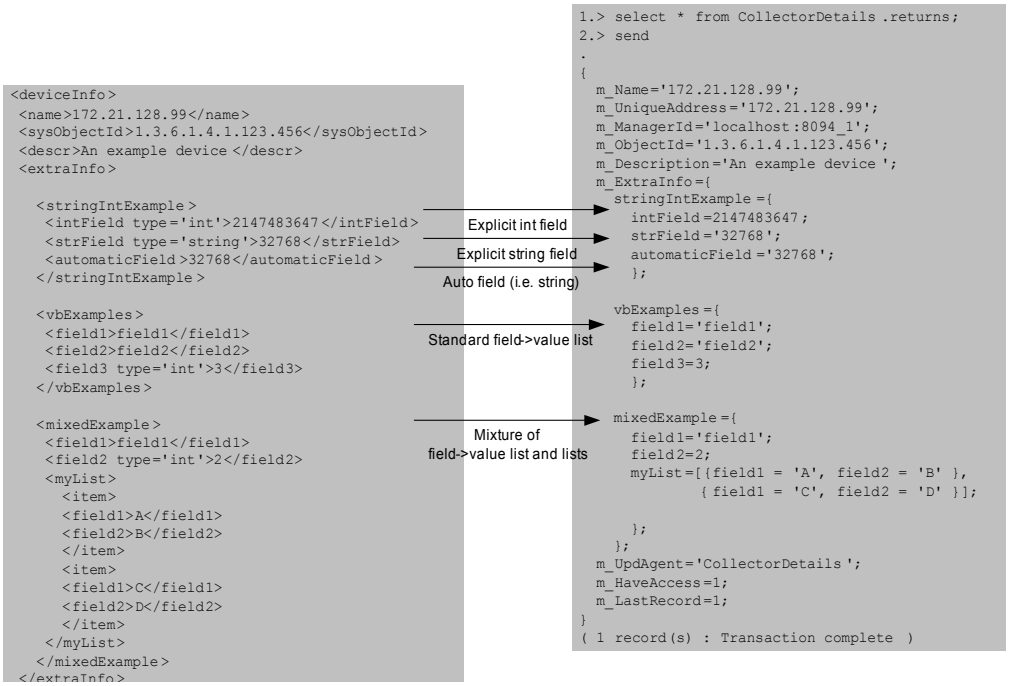
Some of the RPC method responses permit freeform data, which is denoted by the 'extraInfo' XML tag in their XML Schema. Within these tags any XML data will be converted and placed directly into the appropriate Collector agent's return table.

If no *type* attribute is specified, Network Manager interprets all data as strings and only treats lists as lists if there are multiple tags with the same name.

The diagrams below provide an example of how freeform XML tag structures are translated to produce an agent's return table data.



Freeform List Data Example



Freeform Data Example

Chapter 3: Network Manager Collector Support

This section shows how Network Manager interacts with Collectors (i.e. how it acquires and processes Collector data). It is assumed that the user is familiar with the basic Network Manager architecture (finders, helpers, agents, stitchers).

Although Collectors can be developed without the user having much knowledge of this section, the user will benefit from the information presented here because it aids in debugging and testing Collectors.

Collector Relevant Network Manager Processes

Network Manager's support for Collectors is provided by the following processes, which mirror the architecture for the existing Network Manager processes:

Collector Finder (ncp_df_collector)
Reads the Collector host/port seeds from DiscoCollectorFinderSeeds.cfg, and queries the Collector to get a list of managed devices

- XML-RPC Helper (ncp_dh_xmlrpc)
  Relays XML-RPC requests from the Collector supporting agents to Collectors
- Collector Agents
  The following agents support the downloading of data from Collectors;

  CollectorDetails – Retrieves the sysObjectId, sysDescr, name data

  CollectorInventory – Retrieves inventory (local neighbours), entity (ENTITY-MIB style data) and associated addresses (ip table) data

  CollectorLayer3 – Retrieves Layer 3 connectivity data

  CollectorLayer2 – Retrieves Layer 2 connectivity data

  CollectorLayer1 – Retrieves Layer 1 and Microwave connectivity data

  CollectorVpn – Retrieves Layer 2 and Layer 3 VPN data

  CollectorRAN - Retrieves RAN data (see RAN NCIM model), including logical RAN connectivity data

  Custom Perl Agents - Users can issue XML-RPC calls from their custom Perl Agents.

Note: User Defined Custom Perl Agents

Users may wish to write their own custom Perl API based Collector Agents to gather any data from their Collectors that is not explicitly represented in the published XML Schema or that doesn't sit well within the standard XML-RPC calls.

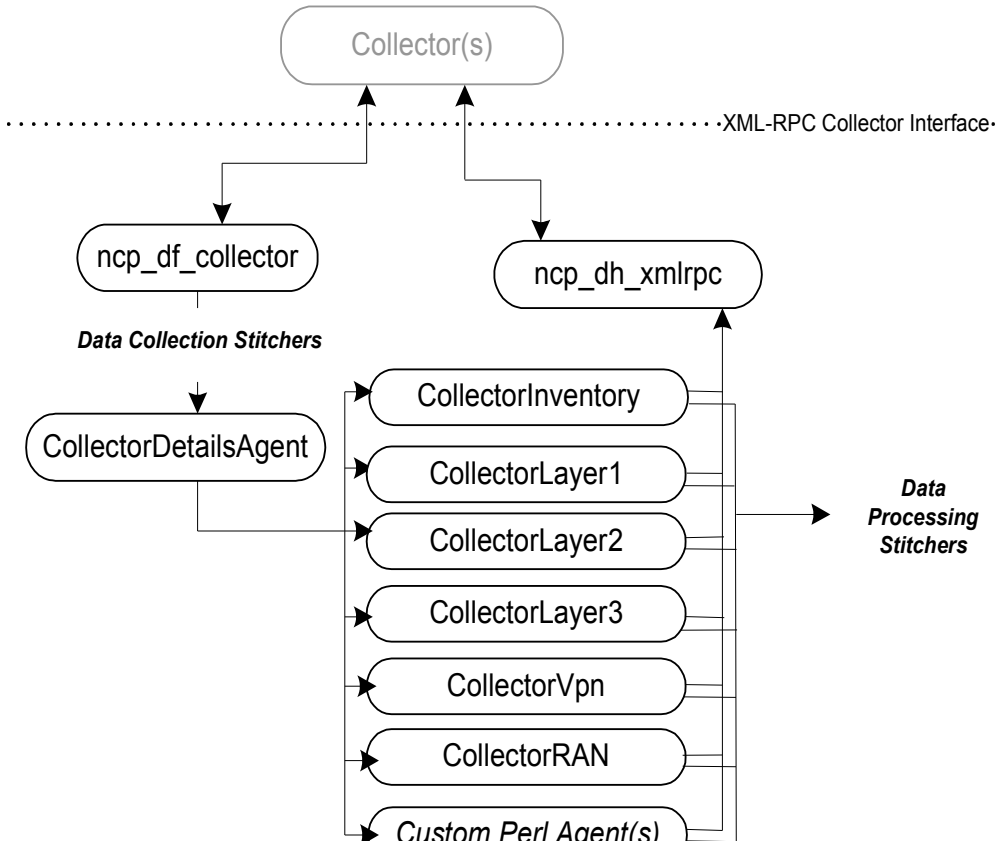
In theory this could also be achieved by adding freeform data to the responses sent by the Collector to the standard Collector Agents, however using custom Agents and custom Collector methods can make the system easier to configure, understand and maintain.

© Copyright IBM Corp. 2014, 2015

A Perl API based Agent can issue any XML-RPC call but it will usually only make sense to issue calls that are not already covered by the existing Collector Agents: This essentially means calls to custom Collector methods via an XML-RPC call. The Perl Agent would then store the result for use in later stitching (possibly custom stitching).

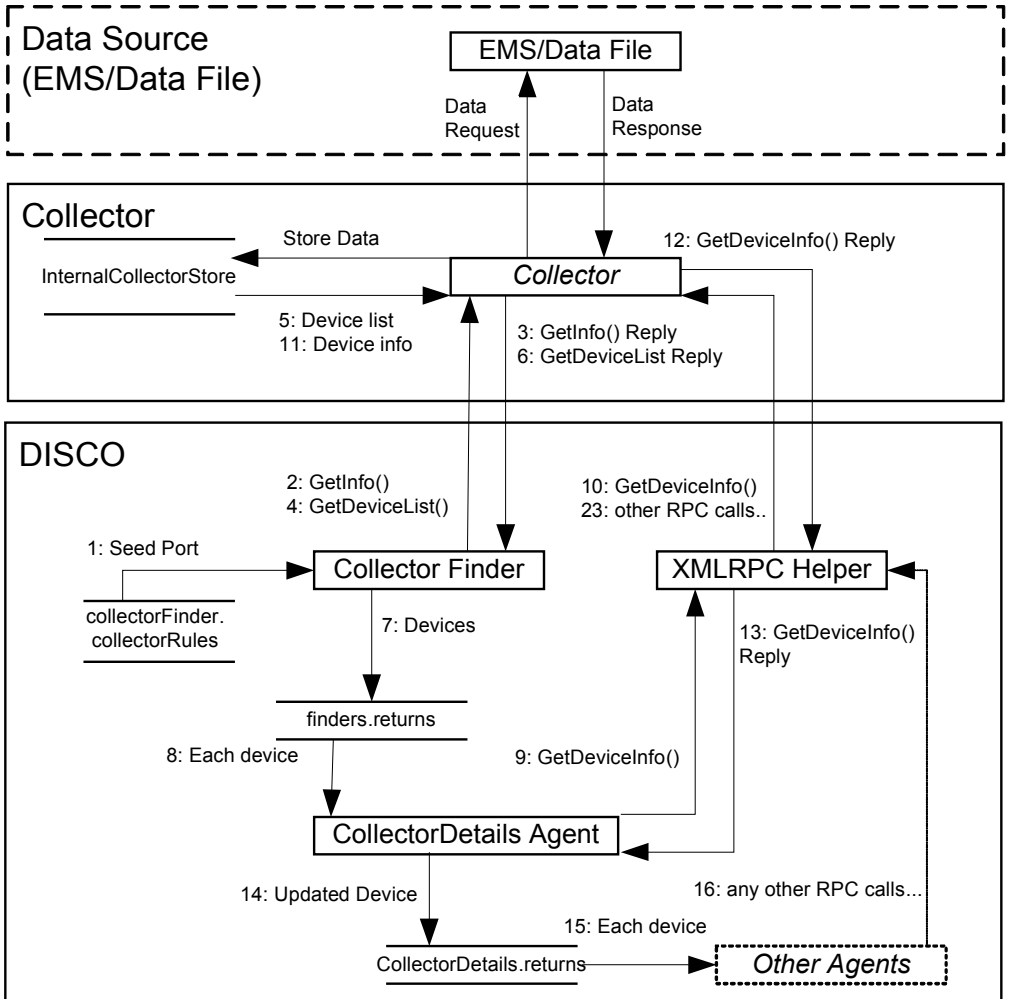
See GetXMLRPCData() and GetXMLRPCEntityData() in the Perl API documentation.

The interaction between these processes is shown in the figure below:



High Level View of Network Manager Collector Support

The sequence of events that occur during a Collector based discovery are shown in the following diagram.



Collector Discovery Sequence

The behavior is the same as in a standard Network Manager discovery, i.e.

- The user configures discovery seed(s)
- The Collector Finder reads the seed(s) and issues an UpdateData() request (to refresh the Collector's data), a GetInfo() request (to get the data source(s) supported by the Collector, and a GetDeviceList() request (to list the devices managed by the Collector)
- The Collector Finder adds all Collector managed devices to finders, returns, noting the managing Collector/data source
- The CollectorDetails agents requests further information on the each device, and passes the updated device records to all Collector supporting agents
- Each Collector supporting agents requests data from the Collector, via the XML-RPC Helper, and outputs records in it's returns table based on the Collector response

- The data from all agents is used by the data processing stitchers to construct the topology

Each process is now described from the perspective of a collector development.

The Collector Finder

Called RPC Methods: GetInfo(), UpdateData(), GetDevice()

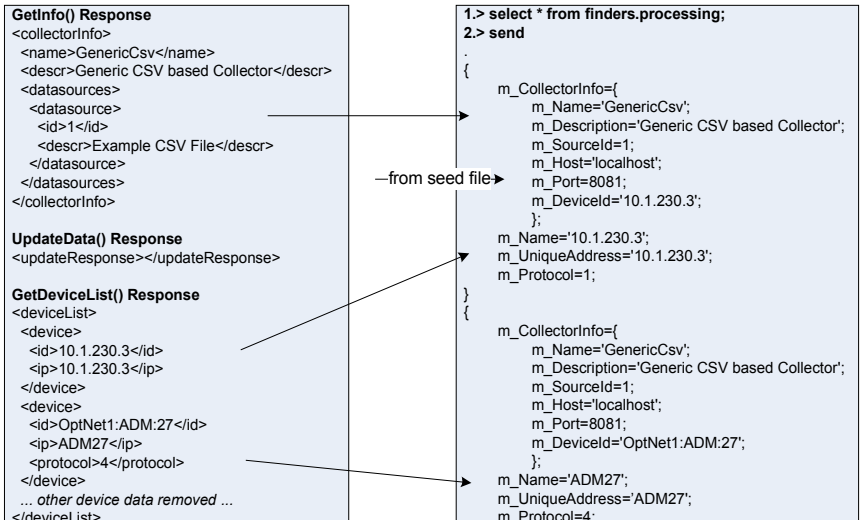
Related Databases: finders.returns / finders.processing

The Collector Finder, ncp_df_collector, is responsible for contracting all Collectors defined in the Collector seed file (DiscoCollectorFinderSeeds.cfg, which contains inserts into the collectorFinder.collectorRules database) and requesting a list of their supported data sources and a list of managed devices for each data source.

The Collector Finder populates the finders.returns table with a record for each managed device. Each record will include an m_CollectorInfo field that specifies the Collector and data source that manage the device.

The Figure below shows how the Collector's XML response maps to the records produced by this process.

Note: records in finders.returns are immediately moved to finders.processing. Therefore, only finders.processing is shown here.



XML to Network Manager Database Mapping

The XML-RPC Helper

Called RPC Methods: relays all agent issued calls

Related Databases: n/a – use debug log output

The XML-RPC Helping, ncp_dh_xmlrpc, enables the Network Manager agents to communicate with the Collectors via XML-RPC using the Network Manager Helper system.

The Helper is configured via DiscoXmlRpcHelperSchema.cfg (xmlRpcHelper.configuration), and DiscoHelperServerSchema.cfg (XmlRpcHelper.XmlRpcHelperConfig table. Debug level 2 is recommended for basic issue investigation.

Note: the Collector Finder does not use this Helper to issue XML-RPC method calls.

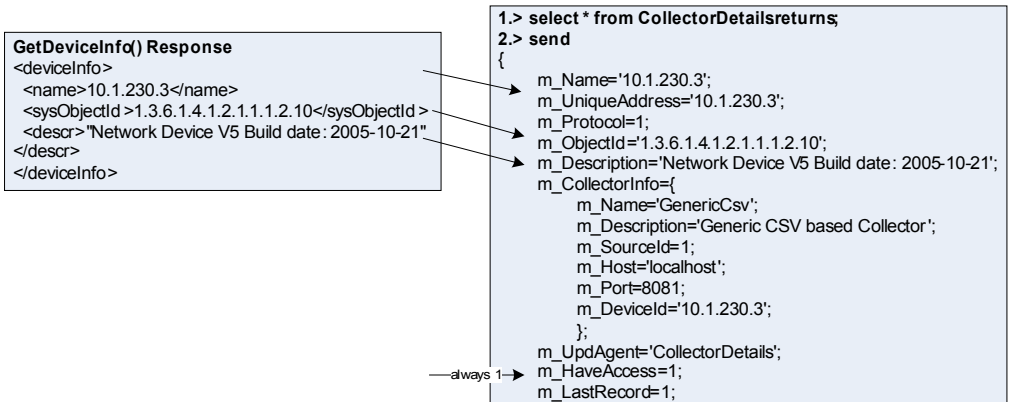
The Collector Details Agent

Called RPC Method: GetDeviceInfo()

Related Databases: CollectorDetails.despatch, CollectorDetails.returns

The Collector Details agent, CollectorDetails, is the equivalent of the SNMP-based Details agent. It is responsible for gathering basic device information to enable Network Manager to decide which agents should process the device.

The agent's fields are populated from the Collector's XML data, as shown below.



Collector XML to Network Manager Database Mapping

The Collector Inventory Agent

Called RPC Methods: GetInventory(), GetEntities(), GetAssociatedAddresses()

Related Databases: CollectorInventory.despatch, CollectorInventory.returns

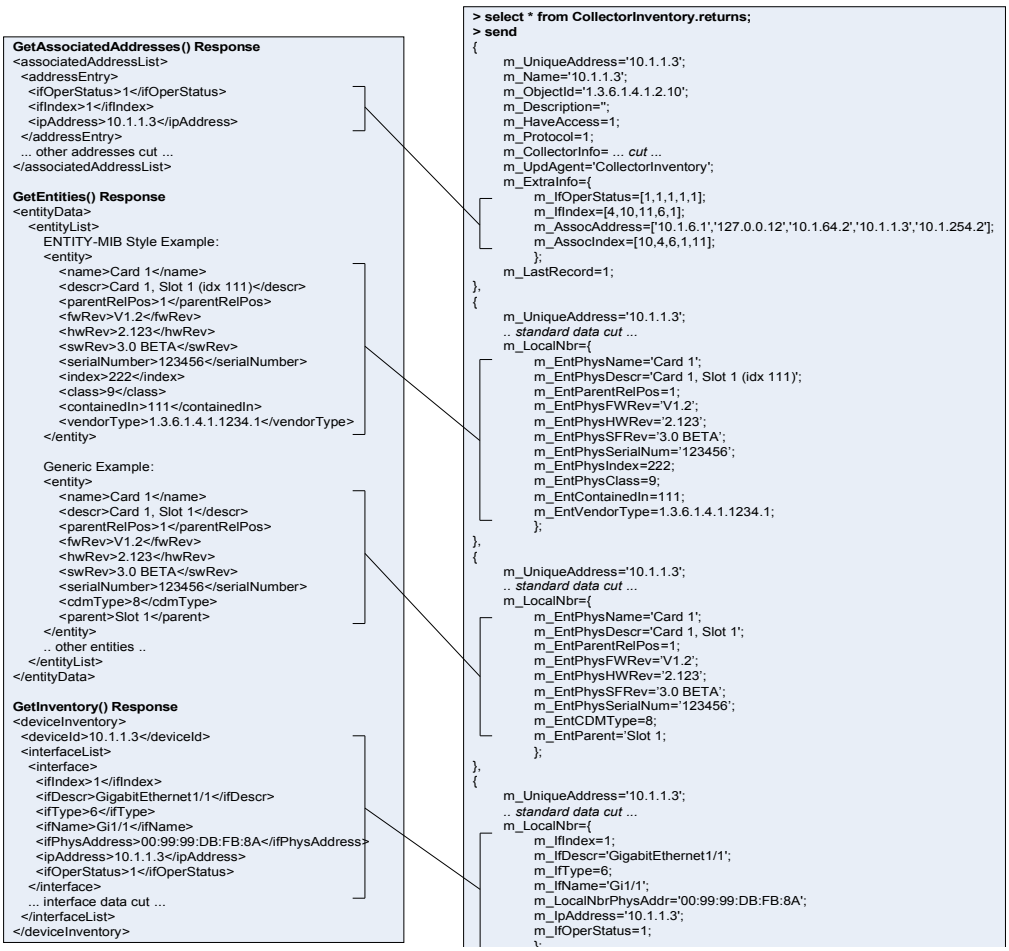
The Collector Inventory agent, CollectorInventory, is the equivalent of the SNMP based Interface, Entity and AssocAddress agents.

It is responsible for gathering the following data:

- Interface information via the GetInventory() call. A new record is created in the returns table for each interface on the device
- ENTITY-MIB style entity information via the GetEntities() call. A new record is created in the returns table for each entity on the device. Note a more flexible generic form of entity information is also supported which relaxes the requirement for entities to have indexes (relationships are represented by names instead) and which supports a CDM entity type rather than class type.

- The IP addresses associated with the device, via the GetAssociatedAddresses() call. Associated address information is only added to the last record in the returns table (the one with m_LastRecord = 1)

The agent's fields are populated from the Collector's XML data as shown in the Figure below.



Collector XML to Network Manager Database Mapping

Note: Where is my GetEntities() Chassis/Interface data going?

Data from the GetEntities() response may be subject to some movement during post-Agent, pre-NCIM Network Manager containment stitching. This is not usually of any interest but when issue investigation requires a step through analysis of the dataflow it can be confusing to the uninitiated.

Data for all the entity types supported by GetEntities() will appear in the m_LocalNbr fields of records in the CollectorInventory.returns table. From

here for all types except Chassis and Port (interface) the data will be copied into the m_ExtraInfo field of new entity records in workingEntities.finalEntity that represent the entity. Finally that data will be transferred to the relevant NCIM table.

For Chassis and Port(interface) entities it is slightly different as data records may already exist thanks to the GetDeviceInfo() and GetInventory() methods (see the note in the XML Schema reference as to why this is). Where possible Network Managers containment stitching attempts to move the Chassis/Port(Interface) data from GetEntities() and merge it with the existing data record.

For chassis records the GetEntities() data will get copied from the m_LocalNbr field of CollectorInventory.returns to the m_ExtraInfo field of the existing Type 1 (main entity) record for the device in workingEntities.finalEntity.

For Ports (interfaces) the containment stitchers attempt to resolve the associated GetInventory() interface data record by matching the interface's RFC1213 ifName or ifDescr to the port's ENTITY-MIB entPhysicalName or entPhysicalDescr. Where the an interface and port cannot not be tied up a new port entity is created for the ENTITY-MIB data to represent the port/interface. In either case the GetEntities() data will be placed in under the m_LocalNbr field (NOT m_ExtraInfo) of the Type 2 (interface) record in workingEntities.finalEntity (wether that be the existing records or newly created one).

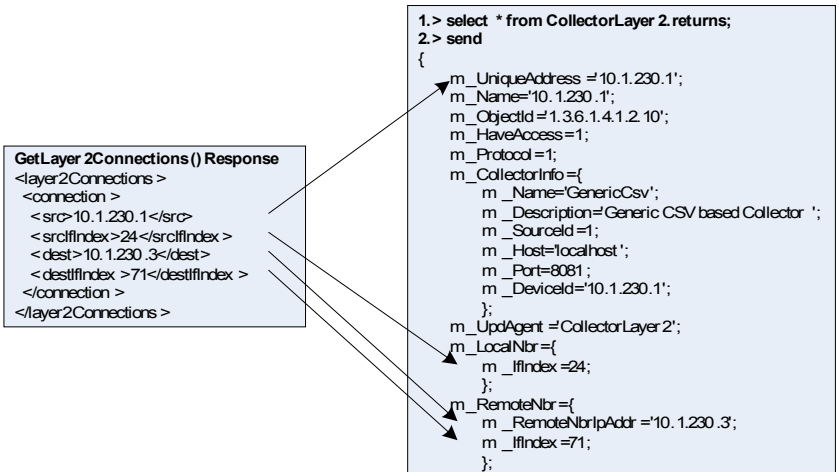
The Collector Layer 2 Agent

Called RPC Method: GetLayer2Connections()

Related Databases: CollectorLayer2.despatch, CollectorLayer2.returns, CollectorSwitchLayer.entityByNeighbor

The Collector Layer 2 agent, CollectorLayer2, is the equivalent of the SNMP-based layer 2 connectivity agents. It is responsible for gathering all Collector layer 2 connectivity data and outputting that data in its returns table as standard Network Manager local and remote neighbor data.

The agent's fields are populated from the Collector's XML data as shown below.



Collector XML to Network Manager Database Mapping

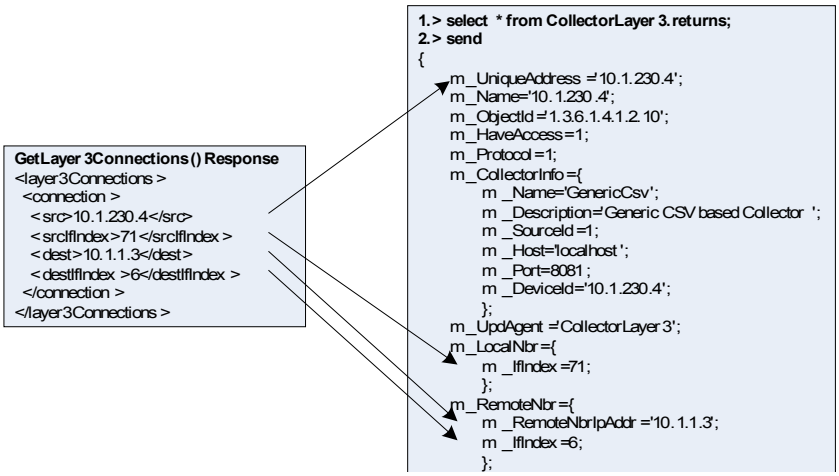
The Collector Layer 3 Agent

Called RPC Method: GetLayer3Connections()

Related Databases: CollectorLayer3.despatch, CollectorLayer3.returns

The Collector Layer 3 agents, CollectorLayer3, is the equivalent of the SNMP-based layer 3 connectivity agents. It is responsible for gathering all Collector layer 3 connectivity data and outputting that data in its returns table as standard Network Manager local and remote neighbor data.

The agent's fields are populated from the Collector's XML data as shown below.



Collector XML to Network Manager Database Mapping

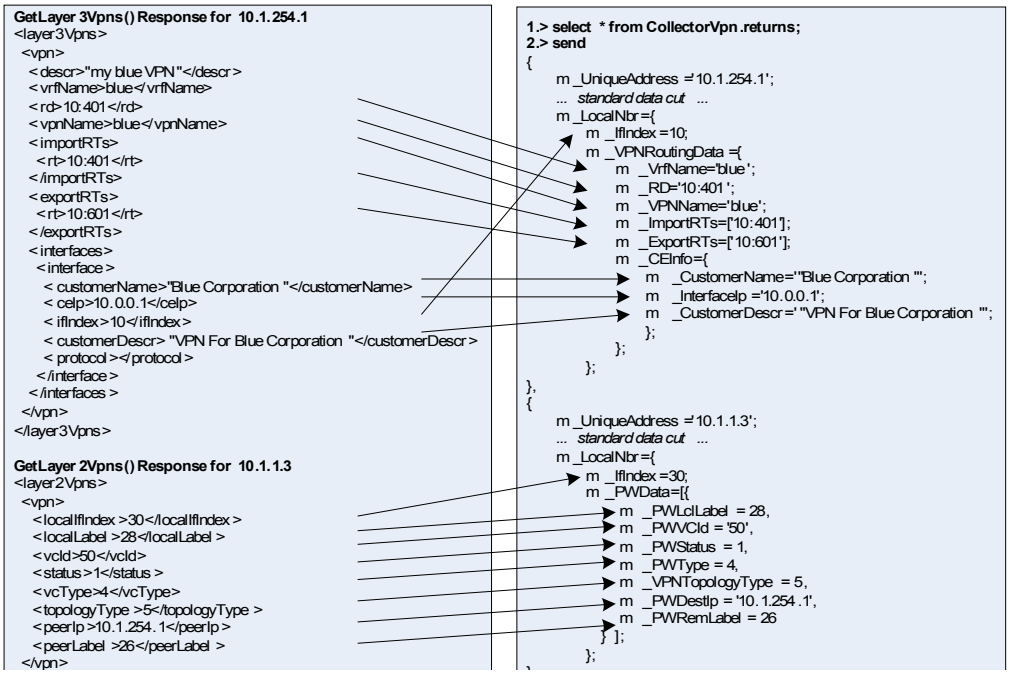
The Collector VPN Agent

Called RPC Methods: GetLayer2Vpns(), GetLayer3Vpns()

Related Databases: CollectorVpn.despatch, CollectorVpn.returns

The Collector VPN agent, CollectorVpn, is the equivalent of the SNMP-based MPLS VPN agents.
The agent is responsible for gathering all Collector VPN data and outputting it in its returns as standard Network Manager VPN data.

The agent's fields are populated from the Collector's XML data as shown below.



Collector XML to Network Manager Database Mapping

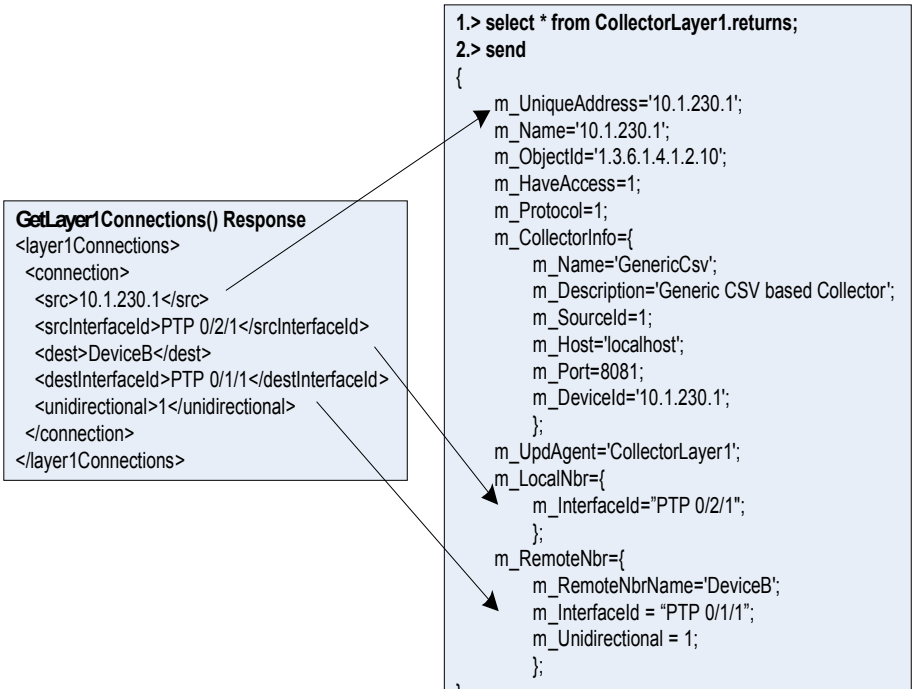
The Collector Layer 1 Agent

Called RPC Method: GetLayer1Connections(), GetConnections(<Microwave>)

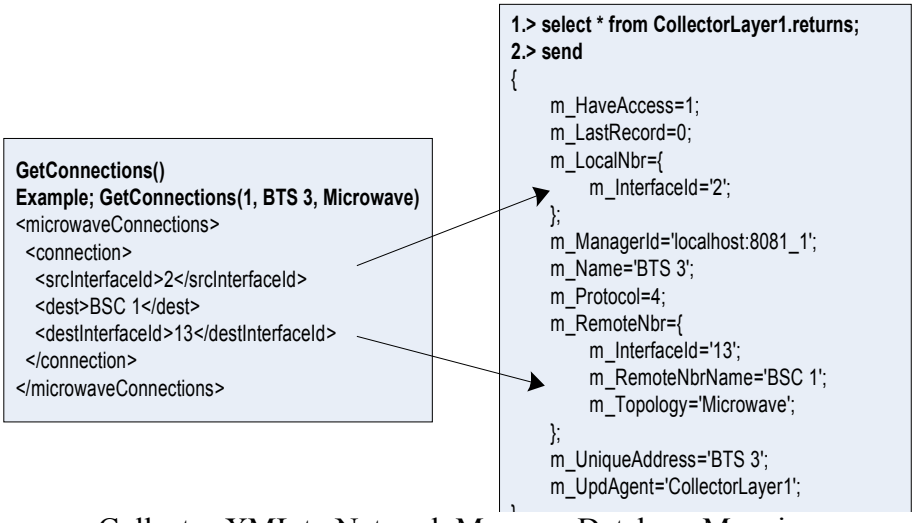
Related Databases: CollectorLayer1.despatch, CollectorLayer1.returns, CollectorL1Layer.entityByNeighbor, MicrowaveLayer.entityByNeighbor

The Collector Layer 1 agent, CollectorLayer1, is responsible for gathering all Collector layer 1 connectivity data and outputting that data in its returns table as standard Network Manager local and remote neighbor data.

The agent's fields are populated from the Collector's XML data as shown below.



Collector XML to Network Manager Database Mapping



Collector XML to Network Manager Database Mapping

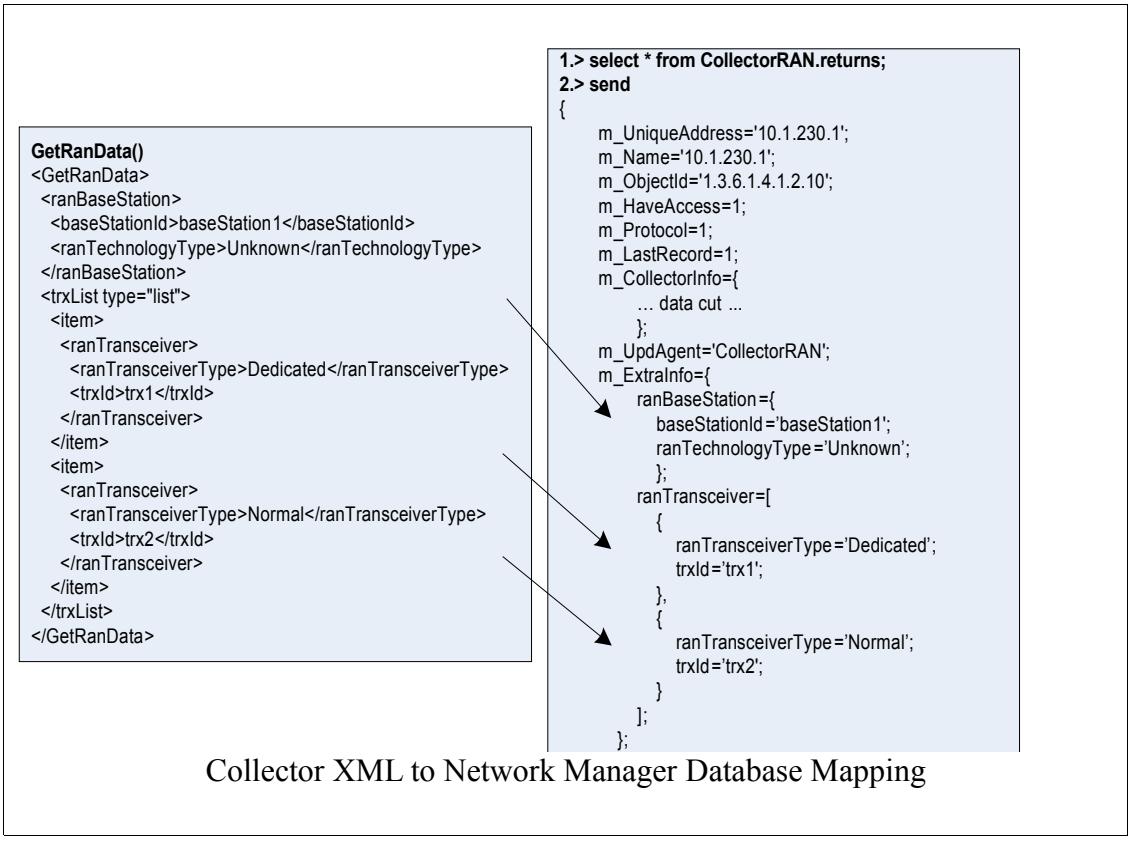
The Collector RAN Agent

Called RPC Method: GetRanData(), GetConnections(<RAN>)

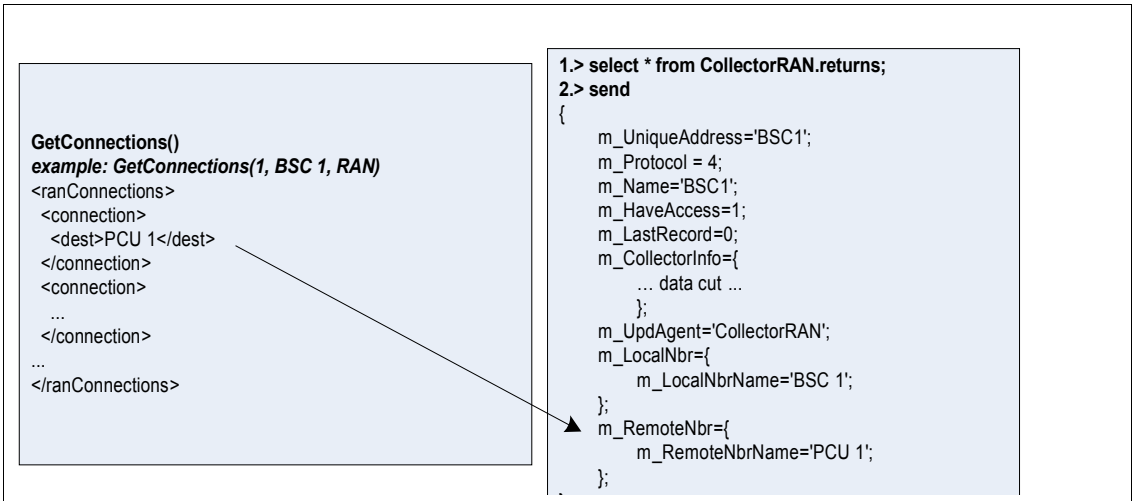
Related Databases: CollectorRAN.despatch, CollectorRAN.returns

The Collector RAN agent, CollectorRAN, is responsible for gathering all Collector RAN data and outputting that data in its returns table. It also gathers logical RAN connectivity data and outputs that data in its returns table as standard Network Manager local and remote neighbor data.

The agent's fields are populated from the Collector's XML data as shown below.



Collector XML to Network Manager Database Mapping



Collector XML to Network Manager Database Mapping

Collector-Relevant Network Manager Data Flow

While the previous section detailed which process databases hold certain data, this section shows how that data flows through those databases during a discovery, enabling a more efficient pinpointing of any issues.

Network Manager's data stitching works in two stages

- Data collection stitching (finders to agents)
- Data processing stitching (agents to scratchTopology)

You are most likely be interested in the data collection stitching when Network Manager does not appear to be discovering anything. If the agents are known to be returning the correct data but the end result in, for example, the GUI is incorrect then you will be more interested in the data processing stitching.

In addition to these stages, there are different modes of discovery that can alter stitching behavior:

- Full Discovery (discover all devices according to the configured seeds and scope)
- Partial Re-discovery (rediscover a subnet or device)

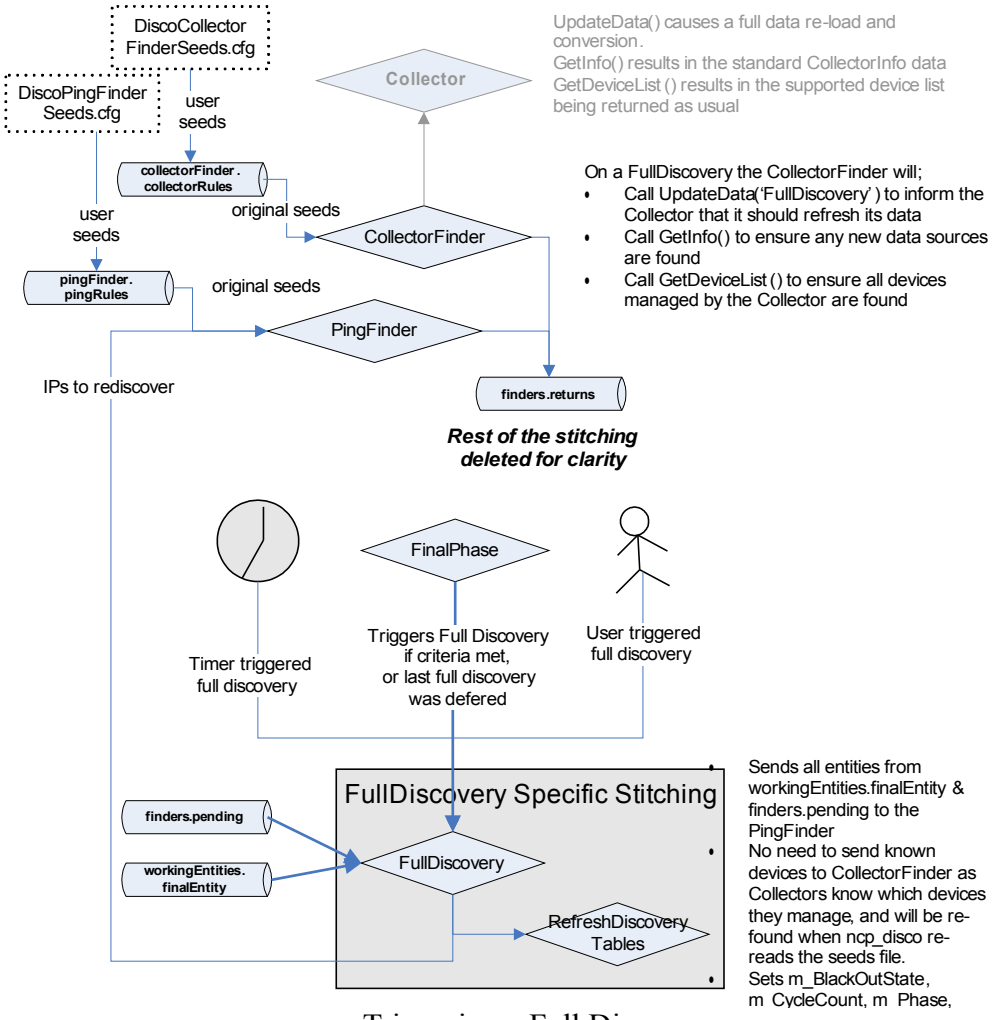
First the data collection and processing stitching for a full discovery will be presented, followed by a section on partial re-discovery, which, for convenience, will cover both partial rediscovery stitching.

It should be noted that the information presented here does not cover every stitcher, nor does it detail everything that the stitchers do; only relevant information is presented.

Full Discovery Mode Stitching

Before presenting the collection and processing stitching, the method of triggering a full discovery is shown.

Full discovery triggering occurs in the data processing stitching and involves a loop back to the finders and initial data collection stitching as shown below.



Triggering a Full Discovery

Once a triggered full discovery has resulted in data being inserted into finders.returns, the data collection stitching is employed to get the records to the agents.
The Figure below shows the most relevant database table stitchers involved in collection stitching. SNMP-based discovery processing has been grayed out.

Key data collection stitching database tables for issue investigations are;

- Finders.processing: has the device been found via a Collector at all?

```
select * from finders.processing
where m_CollectorInfo <> NULL and m_UniqueAddress = <ip>;
```

- CollectorDetails.despatch: has the device passed scoping/detection filter checks?

```
select * from CollectorDetails.despatch
where m_UniqueAddress = <ip>;
```

- CollectorXXX.despatch: has the device been passed to the agents?

```
select * from <agent>.despatch
```

```
where m_UniqueAddress = <ip>;
```

The data processing stitching is responsible for taking data from the agent's returns table and processing it to get connectivity, containment data, and other relationships. The Data processing Figure below shows the stitching performed during data processing stitching during a full discovery.

Key data processing stitching database tables for issue investigation are;

- CollectorXXX.returns: has the device data been discovered?

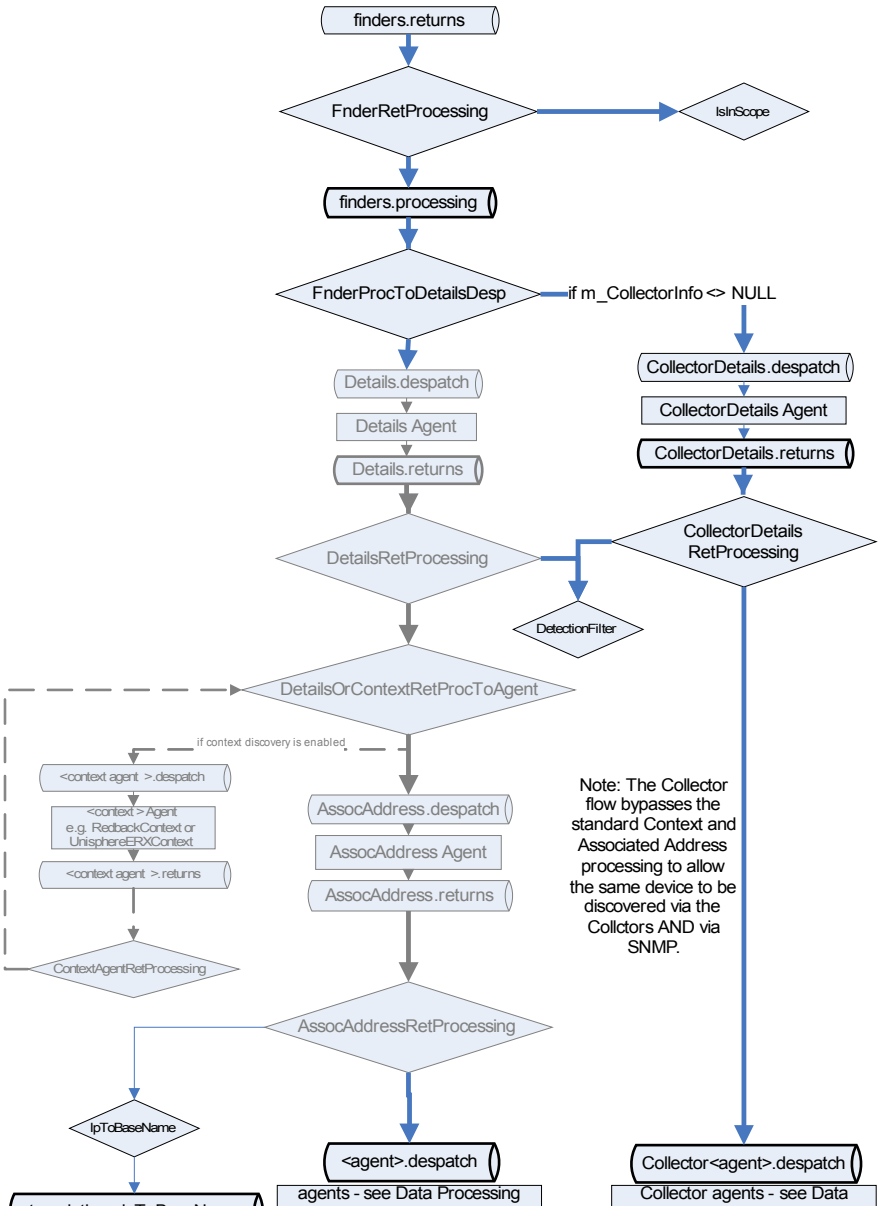
```
select * from <agent>.returns
where m_UniqueAddress = <ip>;
```

- Translations.uniqueAddressToWorkAddress:
- Is Network Manager aware which address is used to represent the device in the Collector agents?
- Is Network Manager aware how this address maps to the main m_WorkAddress?
- Is Network Manager aware how the main m_WorkAddress maps to the m_BaseName used by the later tables?

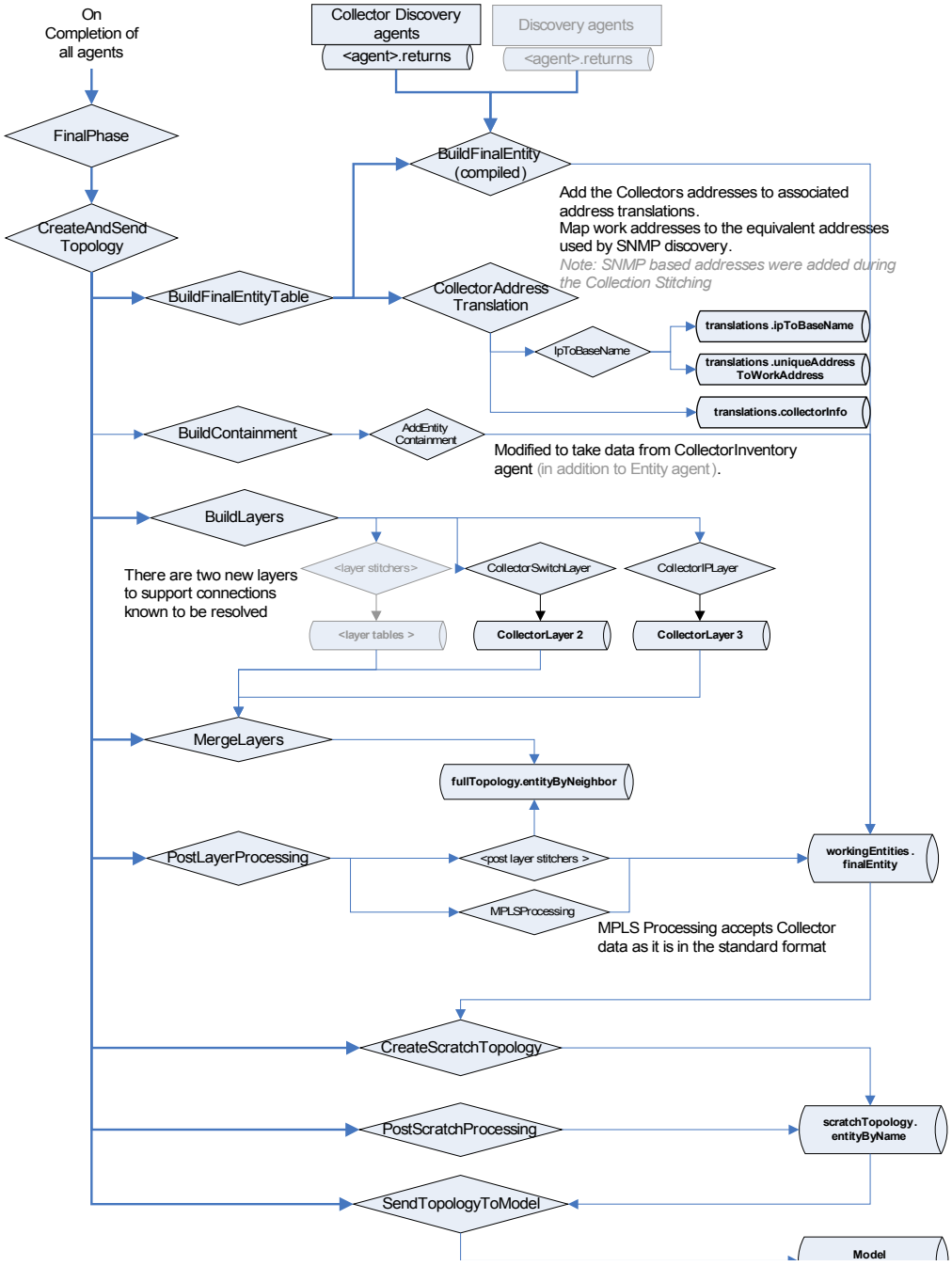
```
select m_WorkAddress from translations.uniqueAddressToWorkAddress
where m_UniqueAddress = <ip>;

select m_BaseName from translations.ipToBaseName
where m_WorkAddress = <work address from earlier query>
```

- scratchTopology.entityByName: has the device made it to the final data table?

```
select * from scratchTopology.entityByName
where m_Name like <base name obtained from last query>;
```



Data Collection Stitching (Full Discovery)



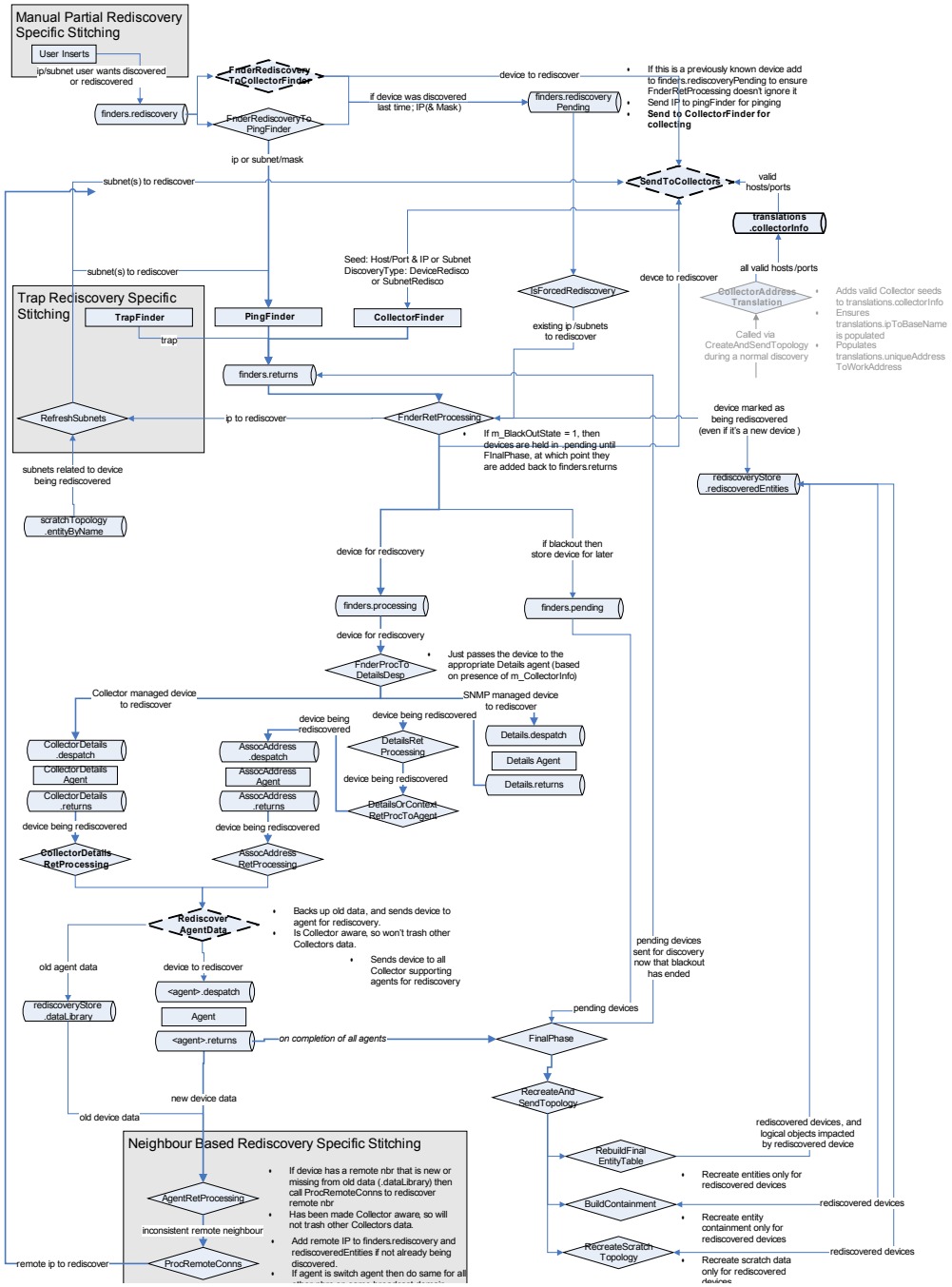
Data Processing Stitching (Full Discovery)

Partial Rediscovery Stitching

Partial rediscovery stitching allows the partial discovery of a network to be merged with an earlier discovery.

The stitching is more complex than during a full discovery and is more easily understood if both the data collection and data processing stitching are presented as one set of stitching.

The Figure below shows the data collection and data processing stitching performed during a partial rediscovery. It also shows the various ways in which a partial rediscovery can be triggered (the grey boxes).



Partial Re-Discovery Stitching

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 2014, 2015

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
958/NH04
IBM Centre, St Leonards
601 Pacific Hwy
St Leonards, NSW, 2069
Australia

IBM Corporation
896471/H128B
76 Upper Ground
London SE1 9PZ
United Kingdom

IBM Corporation
JBFA/SOM1
294 Route 100
Somers, NY, 10589-0100
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, Netcool, Netcool/OMNIbus,and Tivoli are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, and Portable Document Format (PDF) are trademarks or registered trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.