Document Version	V 1.0

CCSP Common Component Software Functional Specification

TR-069 Protocol Agent (TR-069 PA): cloud interface to TR-069 Auto Configuration Server (ACS)

Modification History

Revision	Date	Comments
V 1.0	07/3/2014	Initial Release

Table of Contents

List of Tables

List of Figures

1 Audience

The audience for this document includes:

- Architecture team as reviewers of the Component Architecture Specification
- Design review teams who use this document as a guide to evaluate alignment of design with these functional requirements
- Code review teams who use this document as a guide to evaluate implementation completeness
- Implementers of other component that rely on the interfaces provided by the component described in this document
- Product Instantiation Teams that use this component in point products
- Test Teams will use this document to build test plans and harnesses to exercise the component interfaces

2 Component Overview

Protocol Agents are CCSP components that directly interface to the Cloud. These components facilitate remote administration/management of GW device profiles. The Protocol Agents provides the necessary abstraction to the internal CCSP architecture and internal CCSP components for interacting with the Cloud. The internal CCSP components are not required to be aware of any protocol specific details on the cloud interfaces. As a Protocol Agent could be an SNMP PA or a TR-069 PA..etc.

TR69 ACS uses HTTP/SOAP to communicate with EP or GW. The TR69 protocol agent hides all the protocol specific details and communicates internally using internal namespaces and APIs over the CCSP message bus.

There may be multiple instances of Protocol Agents in a CCSP subsystem.

Protocol Agents perform the following functions:

- Authenticate and establish secure session with their corresponding cloud configuration server during initialization.
- Perform low level protocol handling for downstream traffic.
- Perform low level protocol handling for upstream traffic.
- Routes cloud messages to internal functional components registered for consumption/processing of those namespaces by looking up the Component Registrar.
- “Action” Normalization
    - Internal CCSP components use normalized action / RPC methods to process requests
- Define an XML based Mapping Schema that defines the mapping from external constructs to internal constructs. These constructs will include:

  - External Objects and parameters to Internal Objects and parameters
  - Formatting Conversions
  - Internal Errors to External Errors
- Signals errors and routes responses to corresponding cloud adapters
- Performs all transactions as an atomic operation.
  - For example, if a transaction involves a "SetParameterList" action of 10 key-value pairs, then the Protocol Agent applies the changes to all of the specified key-value pairs atomically. That is, either all of the value changes are applied together, or none of the changes are applied at all.
  - In cases where a single transaction involves multiple Components, the PA aggregates the response from all the components before sending the results back to PA.
- Manages the order of operations within a transaction and across transactions.
  - The PA serializes all transactions from its cloud interface and only allows one transaction at a time.
- Generates and maintains Context for asynchronous notifications and transactions. This is explained in more details in the next section.

3 Design Considerations

- TR-069 PA design must be compliant with CCSP Common Component Design Rules.
- TR-069 PA must be able to receive/handle/response messages from or to the Cloud.
- TR-069 PA must be able to register itself to Component Registrar (CR).
- TR-069 PA must be able to register its own services if any to CR so they can be discovered and requested in the runtime.
- TR-069 PA must be able to discover services provided by other common components if needed.
- TR-069 PA maintains its own configuration settings in the Persistent Storage Manager (PSM). That configuration includes management server related parameters that are defined under "Device.ManagementServer.".
- TR-069 PA loads namespace configuration XML file during startup and manage data model tree.
- TR-069 PA must be able to receive RPC method calls from cloud server (TR-069 ACS), normalize them and request respective functional components (FC) for services, and return results back to cloud server.
- TR-069 PA must be able to receive signals FC delivers regarding parameter value change that is not caused by TR-069 ACS.

3.1 Third Party Relationships

No thirty party software modules will be used.

3.2 Security Considerations

This component implements CPE side of CPE WAN Management Protocol (CWMP) defined by TR-069 specification. This component is totally relying on CWMP to cover all possible security issues and no extra security concerns are expected.

4 Functional Structure and Interface Design

Control Plane architecture abstracts the cloud interfaces for internal CCSP architecture. It makes internal common components like TR-069 PA to be agnostic to those cloud interfaces by using a set of normalized actions.

This component is intended to be responsible for all normalized actions listed below. This component does not provide interface for servicing normalized actions, rather it depends on functional components to expose APIs on CCSP message bus.

- GetParameterNames
- GetParameterValues
- SetParameterValues
- SetCommit
- GetParameterAttributes
- SetParameterAttributes
- AddTblRow
- DeleteTlbRow

4.1 Data Model and Structures

North bound interface to cloud, this component implements CWMP CPE. Through south bound interface, this component interacts with CR and FCs to perform normalized actions. And the related data structures and function prototypes will be defined by CCSP Architecture Specification, and please refer to it for details. This document only discusses specific items related to the TR-069 PA component.

Upon start up, the TR-069 PA loads a mapping configuration file which includes information on how to map external namespaces specific to the PA to internal namespaces on supported data model parameters as well as external and internal data types of each parameter. The schema of the configuration file is to be defined, and will be included in Software Design Document. This specification does not intend to define it. The mapping file is further described in section 5.1.1.

With respect to downloadable applications that define their own data model (defining external data model only), TR-069 PA must be able to route normalized calls to appropriate applications via Application Framework Adapter which should be covered by TR-069 Protocol Agent Software Design Document. The Application Framework Adapter is the default route for all data model namespace for which there is no explicit mapping to an internal namespace.

4.1.1 TR-069 PA mapping file schema

TR-069 PA needs a XML mapping file to achieve the following goals.

- Error Mapping
  CCSP Component Design Rules specification defines some CCSP generic error codes. As well, each functional component may define its own CCSP error codes. TR-069 PA has to maintain the mapping between CCSP error codes and TR-069 error codes. The goal is to make CCSP components agnostic to any protocol specific error codes and data models.

- Non-normalized RPC and CCSP namespace mapping
  For some TR-069 RPCs that are not normalized, such as Download, Upload, ChangeDUState, under current architecture specification, these RPCs are implemented through CCSP internal namespaces. Therefore TR-069 PA must be able to map RPC name and all arguments into CCSP internal namespaces. PA is depending on functional component specification to define the mapping. For Download, Secure Software Download Specification is referred. For ChangeDUState, Application Environment Manager functional spec defines CCSP internal namespaces.

  However considering the complexity and "arbitrary" definitions of CCSP internal namespaces for those RPCs, it may turn out PA may have to ignore this mapping and use hard-coded mapping logic in PA. The reason is we do not make PA too complicated for something simply will not change.

  RPC mapping schema is defined for possible future use. This specification does not enforce using RPC namespace mapping unless all mappings are straightforward and in the way that can be managed fairly easy by PA.

  Current implementation of TR-069 PA simply ignores RPC mapping if there's any.

- TR-069 data type and CCSP data type mapping
  The internal CCSP data model is based on TR-181 and follows the data types as

defined in TR-181. Therefore it is not necessary to map data types since the TR69 ACS also uses TR-181 for configuration and diagnostics. The optional data type mapping is only for simply mapping. In foreseeable future, TR-069 PA is not expected to implement any data type mapping due to the significant differences CCSP and cloud data types, value range, the way to interpret them. We list this item here for completeness of discussion.

- Namespace mapping
  It is required by the current architecture design that the TR-069 PA CCSP component needs to list all supported TR parameters. The mapping file contains the internal data model parameters supported by all CCSP components and their corresponding external cloud names. For TR-69, however this would be an easy mapping since the internal CCSP Data model is based on TR-181.
  Also considering how complexity a CPE may be, TR-069 PA may have to know which sub-system a parameter is supposed to be retrieved from. Due to complexity of data type mapping, no data type/value mapping is needed.

  To support use case of multiple sub-systems, each parameter listed includes an optional sub-system name to indicate that the owning Functional Component is on the specified sub-system. TR-069 PA therefore queries local CR to discover that owning FC.

  Moreover, the mapping file also includes CCSP internal namespaces that are invisible to the cloud server through TR-069 PA. Usually those namespaces are extensions to TR data models, but this is not required, meaning even for TR data model namespaces, we can do exactly the same if it makes sense.

  When PA receives a GetParameterNames request on “Device.” for example, PA through the mapping file and knows the possible sub-systems, and queries local CR to discover all relating FCs that own namespaces under “Device.”. Then PA makes a GetParameterNames normalized action call to each owning FC on local or remote sub-system. After the call returns, PA needs to check each returned namespace against the mapping file to filter out un-supported or invisible ones, and then merge results and sends to cloud server.

The following list shows the XML schema for TR-069 PA mapping file.

Table 1 TR-069 PA mapping file schema

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

  <!-- define error mappings from CCSP errors to TR-069 CPE errors -->
  <xs:complexType name="ErrorMapping">
    <xs:sequence>
      <xs:element name="CcspError" type="xs:integer" />
      <xs:element name="Tr069CpeError" type="xs:integer" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ErrorMapper">
    <xs:sequence>
      <!-- CPE error mapping entries -->
      <xs:element name="Error" type="ErrorMapping" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>


  <!-- RPC mapper -->

  <!-- argument that has no mapping namespace does not have to be listed -->
  <xs:complexType name="RpcArgMapping">
    <xs:sequence>
      <!-- Namespace is just the last name  -->
      <xs:element name="ArgName" type="xs:string" />
      <xs:element name="Namespace" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="FileTransferMapper">
      <xs:sequence>
          <xs:element name="ArgMapper" type="RpcArgMapping"
            minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="CcspPathName" type="xs:string" use="required"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="NoArgRpcMapper">
    <xs:sequence>
      <xs:element name="Namespace" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ChangeDUStateMapper">
      <xs:sequence>
          <xs:element name="ArgMapper" type="RpcArgMapping"
              minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="CcspPathName" type="xs:string" use="required"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="DUStateChangeCompleteMapper">
      <xs:sequence>
          <xs:element name="ArgMapper" type="RpcArgMapping"
              minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TransferCompleteMapper">
      <xs:sequence>
          <xs:element name="ArgMapper" type="RpcArgMapping"
              minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
  </xs:complexType>

  <xs:complexType name="RpcMapper">
    <xs:sequence>
      <xs:element name="Download" type="FileTransferMapper" minOccurs="0" />
      <xs:element name="Upload" type="FileTransferMapper" minOccurs="0" />
      <xs:element name="Reboot" type="NoArgRpcMapper" minOccurs="0" />
      <xs:element name="FactoryReset" type="NoArgRpcMapper" minOccurs="0" />
      <xs:element name="ChangeDUState" type="ChangeDUStateMapper" minOccurs="0" />
      <xs:element name="DUstateChangeComplete" type="DUStateChangeCompleteMapper" minOccurs="0" />
      <xs:element name="TransferComplete" type="TransferCompleteMapper" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>



  <!-- ValueChange to RPC mapper -->
  <xs:complexType name="AcsRpcEventType">
    <!--  CcspPathName refers to the object that contains namespace that is used to monitor operation state change -->
   <xs:attribute name="CcspPathName" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="MonitorName" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="SuccessState" type="xs:string" use="required"></xs:attribute>
   <xs:attribute name="FailureState" type="xs:string" use="required"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="VCRpcMapper">
    <xs:sequence>
      <xs:element name="DownloadComplete" type="AcsRpcEventType" minOccurs="0" />
      <xs:element name="UploadComplete" type="AcsRpcEventType" minOccurs="0" />
      <xs:element name="DUStateChangeComplete" type="AcsRpcEventType" minOccurs="0"  />
    </xs:sequence>
  </xs:complexType>


  <!-- list of supported parameters -->
  <xs:simpleType name="ParamType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="string"/>
      <xs:enumeration value="int"/>
      <xs:enumeration value="uint"/>
      <xs:enumeration value="bool"/>
      <xs:enumeration value="dateTime"/>
      <xs:enumeration value="base64"/>
      <!-- perhaps we don't need to support 'anySimpleType' defined by TR-069, we will cover that if necessary -->
    </xs:restriction>
  </xs:simpleType>

  <!-- CCSP param types -->
  <xs:simpleType name="CcspParamType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="string"/>
      <xs:enumeration value="int"/>
      <xs:enumeration value="uint"/>
      <xs:enumeration value="bool"/>
      <xs:enumeration value="dateTime"/>
      <xs:enumeration value="base64"/>
      <xs:enumeration value="float"/>
      <xs:enumeration value="double"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- parameter info -->
  <xs:complexType name="ParamInfo">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" />
      <xs:element name="Type" type="ParamType"   minOccurs="0" />
      <xs:element name="Writable"  type="xs:boolean"  minOccurs="0" />
      <!--  SubSystem is used to indicate which subsystem the FC resides that owns the namespace -->
      <xs:element name="SubSystem"  type="xs:string"  minOccurs="0" />
      <!-- define CcspNamespace and CcspParamType for completion, they will be ignored by TR-069 PA -->
      <xs:element name="CcspNamespace" type="xs:string" minOccurs="0" />
      <xs:element name="CcspParamType" type="CcspParamType" minOccurs="0" />
    </xs:sequence>
    <!—optional attribute to indicate if the namespace is not visible to cloud server -->
    <xs:attribute name="invisible" type="xs:boolean" use="optional" default="false"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="ParamList">
    <xs:sequence>
      <xs:element name="Param" type="ParamInfo"  minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- define supported data model -->
  <xs:complexType name="DataModelType">
    <xs:sequence>
      <xs:element name="URL" type="xs:string" />
      <xs:element name="URN" type="xs:string" />
      <xs:element name="Features" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SupportedDataModelType">
    <xs:sequence>
      <!-- CPE error mapping entries -->
      <xs:element name="DataModel" type="DataModelType" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- root node of mapper file -->
  <xs:element name="Tr069Mapper">
    <xs:complexType>
      <xs:all>
        <xs:element name="ErrorMapper" type="ErrorMapper" />
        <xs:element name="RpcMapper" type="RpcMapper" />
        <xs:element name="VCRpcMapper" type="VCRpcMapper" />
        <!-- supported parameter list can be retreived from CR, conceptually defined in mapper file for completion -->
        <xs:element name="ParamList" type="ParamList" minOccurs="0" />
        <xs:element name="SupportedDataModel" type="SupportedDataModelType" />
      </xs:all>
    </xs:complexType>
  </xs:element>

 </xs:schema>
```

Below is a sample of TR-069 PA mapping file.

Table 2 Sample TR-069 PA mapping configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<Tr069Mapper>
   <ErrorMapper>
      <Error>
          <CcspError>1000</CcspError>
          <Tr069CpeError>9000</Tr069CpeError>
     </Error>
     <Error>
          <CcspError>1001</CcspError>
         <Tr069CpeError>9001</Tr069CpeError>
      </Error>
      ……
   </ErrorMapper>
   <RpcMapper>
      <Reboot>
          <Namespace>CCSP.COMMAND.REBOOT</Namespace>
      </Reboot>
      <FactoryReset>
          <Namespace>CCSP.COMMAND.FACTORYRESET</Namespace>
     </FactoryReset>
      <Download CcspPathName=”SWDownload.{i}.”>
          <ArgMapper>
             <ArgName>CommandKey</ArgName>
            <Namespace>DownloadTransferID</Namespace>
        </ArgMapper>
        <ArgMapper>
             <ArgName>FileType</ArgName>
            <Namespace>ImageType</Namespace>
        </ArgMapper>
        <ArgMapper>
             <ArgName>URL</ArgName>
            <Namespace>DownloadSrcURI</Namespace>
        </ArgMapper>
        <ArgMapper>
             <ArgName>Username</ArgName>
            <Namespace>DownloadUsername</Namespace>
        </ArgMapper>
        <ArgMapper>
```

```
            <ArgName>Password</ArgName>
           <Namespace>DownloadPassword</Namespace>
        </ArgMapper>
        <ArgMapper>
            <ArgName>FileSize</ArgName>
           <Namespace>ImageSize</Namespace>
        </ArgMapper>
        …
      </Download >

      <ChangeDUState CcspPathName="CCSP.COMMAND.PACKAGE.MANAGEMENT.{i}." >
        <ArgMapper>
            <ArgName>URL</ArgName>
            <Namespace>URL</Namespace>
        </ArgMapper>
            <ArgName>UUID</ArgName>
            <Namespace>UUID</Namespace>
        </ArgMapper>
        …
      </ChangeDUState>
      <DUStateChangeComplete>
        <ArgMapper>
            <ArgName>CommandKey</ArgName>
            <Namespace>ID</Namespace>
        </ArgMapper>
            <ArgName>UUID</ArgName>
            <Namespace>UUID</Namespace>
        </ArgMapper>
        …
      </ DUStateChangeComplete>

       …
    </RpcMapper>
    <ParamList>
       <Param>
          <Name>Device.DeviceInfo.Manufacturer</Name>
          <Type>string</Type>
          <Writable>0</Writable>   <!—if a parameter is read-only, this tag can be absent -->
          <!-- CCSP name and type conversion not expected, so even though CcspNameSpace or
              CcspParamType is listed, TR-069 PA is going to ignore it. They are defined solely for
              possible future discussion.
          -->
       </Param>
       …
    </ParamList>
    <!– ValueChange to ACS RPC mapping -->
```

```
  <VCRpcMapper>
    <DownloadComplete CcspPathName=”SWDownload.{i}.”
      MonitorName=”ProcessingState”
      SuccessState=”Success”
      FailureState=”Fail”
      />
    <DUStateChangeComplete CcspPathName=”CCSP.COMMAND.PACKAGE.MANAGEMENT.{i}.”
      MonitorName=”OperationState”
      SuccessState=”Complete”
      FailureState=”Error”
      />
  </VCRpcMapper>
</Tr069Mapper>
```

4.1.2 CCSP Message Bus APIs

Per CCSP Architecture specification and CCSP Common Component Design Rules document, we will only define message bus APIs for the following normalized actions.

- GetParameterNames
- GetParameterValues
- SetParameterValues
- GetParameterAttributes
- SetParameterAttributes
- SetCommit
- AddTblRow
- DeleteTblRow

4.1.3 CCSP namespaces defined for other normalized actions

For other normalized actions, they will be covered with data model approach. In details, CCSP namespaces are defined for each action. These actions will be requested by PA via setting their values respectively to the namespaces defined. These namespaces are defined in each Functional Component Software Architecture Specification.

CCSP Common Component Design Rules document does not define a specific action and let a FC to do it, the mapping must be included into TR-069 PA namespace mapping file. For action such as Download, depending on our methodology, we may require only single FC to handle all download, or we allow multiple FCs to handle different type of downloads. For latter case, the namespace mapping file must include a mapping entity per download file type. With current CCSP architecture, single functional component will take care of download of all file types. Please refer Secure Software Download Software Architecture Specification for details.

4.1.4 CCSP data model related events

The following data model related events must be handled by TR-069 PA.

- DUStateChangeComplete*
- AutonomousDUStateChangeComplete*
- TransferComplete**
- AutonomousTransferComplete**
- ValueChanged

* must be supported if CPE supports normalized action 'ChangeDUState'.

** must be supported if CPE supports Download or Upload.

Event ValueChanged needs to be examined carefully by TR-069 PA. FC signals ValueChanged event on message bus on a parameter that the notification counter is larger than zero. Also FC indicates which component causes the value change by specifying 'writeID' which was used in SetParameterValues and SetCommit actions.

In order for TR-069 PA to capture event ValueChanged, TR-069 PA must subscribe to the signal through Component Registrar. And once TR-069 PA receives such event, it needs to check if the value change is caused by itself by checking 'writeID' included in the signal against TR-069 PA's own writeID. If they are same, this signal is simply ignored. Otherwise, PA needs to either notify ACS about a parameter value change in active or passive way as defined by TR-069 specification, or for some special namespaces, PA knows they are asynchronous event indication, could it be TransferComplete, DUStateChangeComplete, or diagnostics completion and so on.

4.2 Non-Data Model APIs

4.2.1 CCSP Message Bus APIs

Component Registration is the same for all common components. We will use whatever defined in CR software design specification to register TR-069 PA and subscribe signals this component wants to capture such as ParameterValueChangeSignal.

You may refer to Component Registrar SFS, SDS, and CCSP High Level Architecture Specification for function prototypes related to Message Bus APIs.

TR-069 PA must follow transaction control mechanism defined by CCSP Architecture specification based on session ID. For details please refer section 7 Session Integrity in

In case normalized action call on external namespaces that do not have counterpart internal namespaces or say PA cannot get such mapping information from CR, TR-069 PA must be able to route calls to default namespace handler which could be Application Environment Manager through private interface per CCSP Component Registrar specification.

4.2.2 Vendor Configuration File Download

Required by the current architecture design, TR-069 PA is required to download and apply Vendor Configuration File through TR-069 Download RPC method.

The configuration file contains namespace/value pairs in clear text or encoded format (TBD if encoding is needed). When TR-069 PA gets a Download RPC method call and file type is specified as "3 Vendor Configuration File", PA makes a request to Secure Software Download (SSD) CCSP component, and after SSD download has succeeded, PA reads configure file content and make one or more SetParameterValues to the owning CCSP Functional Components to apply configuration data. At last, PA makes ACS RPC method TransferComplete to indicate Download has succeeded or failed.

In case service providers require exporting Vendor Configuration File from CPE and later on importing it to the CPE, this approach cannot cover it. In such case, we may have to define a file encoding schema to differentiate all possible configuration file formats. We will revisit this topic when this comes into reality.

Figure 6 illustrates the sequence of this operation.

4.2.3 Common HAL

N/A

4.2.4 Component Specific HAL

N/A

4.3 Portability Requirements

This component must be implemented as platform independent. Therefore there is no effort regarding porting it to different hardware platforms.

5 Component Behavior and System Flow

Figure 1 shows the overview of TR-069 Protocol Agent.

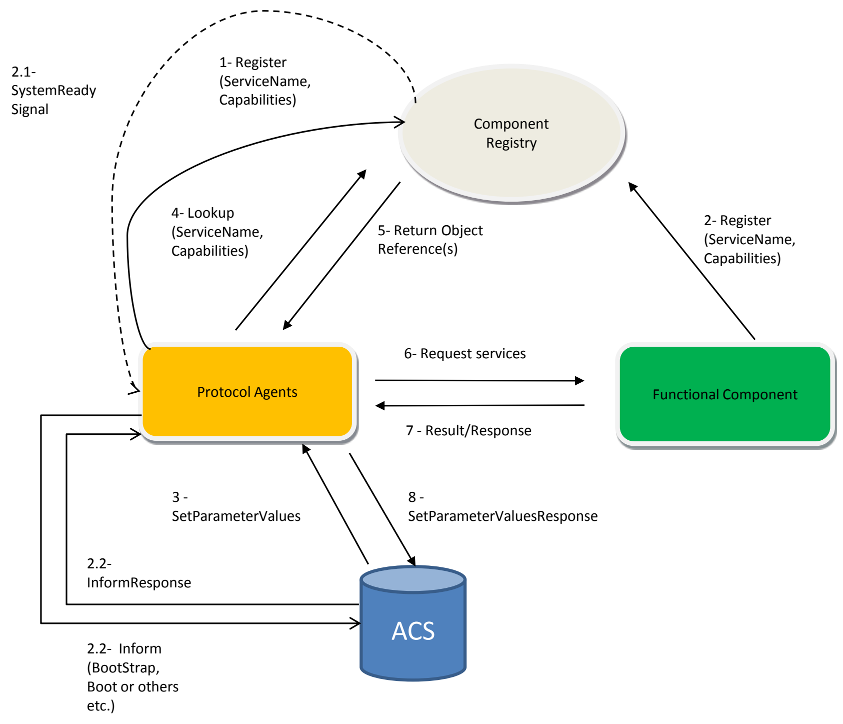
2.1- SystemReady Signal
1- Register (ServiceName, Capabilities)
Component Registry
4- Lookup (ServiceName, Capabilities)
5- Return Object Reference(s)
2- Register (ServiceName, Capabilities)
6- Request services
Protocol Agents
Functional Component
7 - Result/Response
3 - SetParameterValues
8 - SetParameterValuesResponse
2.2- InformResponse
ACS
2.2- Inform (BootStrap, Boot or others etc.)

Figure 1 CCSP TR-069 PA System Overview

This component provides cloud interface to TR-069 ACS and get services from internal CCSP interface to communicate with functional components via normalized actions and internal namespaces.

As you can see from figure 1, when TR-069 PA receives a request from ACS, it resolves namespace with CR, and makes a normalized action invocation over message bus into target functional component. Once TR-069 PA receives response or error from functional component, it construct CWMP SOAP message back to ACS.

Figure 2 demonstrates the data flow. It shows how TR-069 PA receives a RPC request from TR-069 ACS and converts it to one or more normalized action calls into functional component(s).

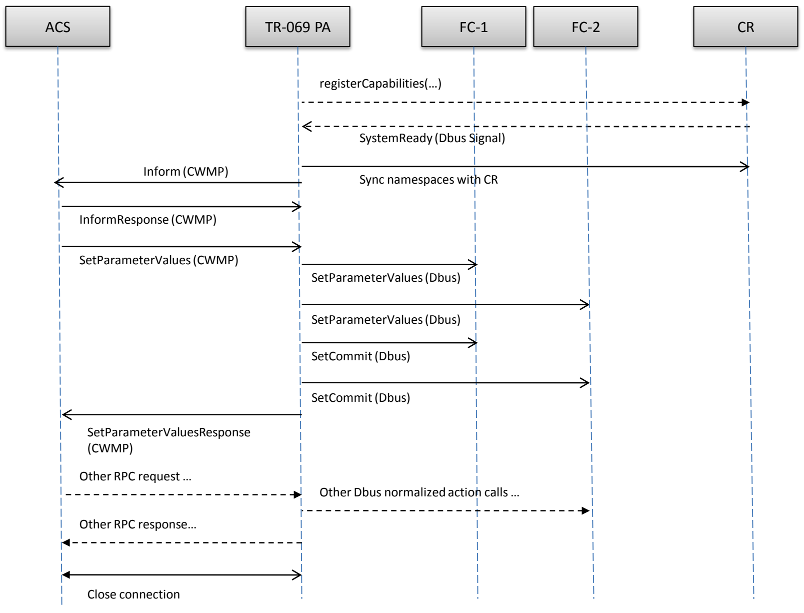
ACS
TR-069 PA
FC-1
FC-2
CR
registerCapabilities(...)
SystemReady (Dbus Signal)
Inform (CWMP)
Sync namespaces with CR
InformResponse (CWMP)
SetParameterValues (CWMP)
SetParameterValues (Dbus)
SetParameterValues (Dbus)
SetCommit (Dbus)
SetCommit (Dbus)
SetParameterValuesResponse (CWMP)
Other RPC request ...
Other Dbus normalized action calls ...
Other RPC response...
Close connection

Figure 2 TR-069 PA SetParameterValues data flow

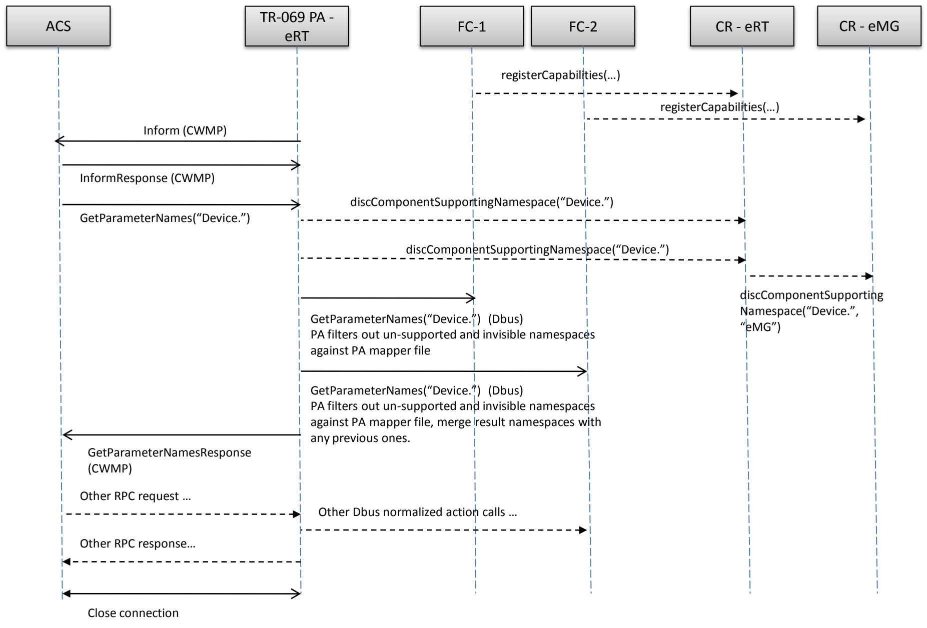
ACS
TR-069 PA - eRT
FC-1
FC-2
CR - eRT
CR - eMG
registerCapabilities(...)
registerCapabilities(...)
Inform (CWMP)
InformResponse (CWMP)
GetParameterNames("Device.")
discComponentSupportingNamespace("Device.")
discComponentSupportingNamespace("Device.")
discComponentSupporting Namespace("Device.", "eMG")
GetParameterNames("Device.") (Dbus)
PA filters out un-supported and invisible namespaces against PA mapper file
GetParameterNames("Device.") (Dbus)
PA filters out un-supported and invisible namespaces against PA mapper file, merge result namespaces with any previous ones.
GetParameterNamesResponse (CWMP)
Other RPC request ...
Other Dbus normalized action calls ...
Other RPC response...
Close connection

Figure 3 TR-069 PA GetParameterNames data flow

Figure 3 demonstrates the data flow when ACS invokes RPC

GetParameterNames("Device."). PA needs to find out all owning FCs on local and remote sub-systems, and make GetParameterNames normalized action calls. Once it gets results, PA needs to filter out un-supported and invisible namespaces (no matter partial or full names) and send back to ACS.

Figure 4 demonstrates the control and data flow on how Software Secure Download can possibly performed. Note that this diagram is only for demonstration purpose and may not reflect actual software design. Final design will be approved by architecture team and this is just an option for them to consider. The design goal presented here is to make TR-069 PA as less coupled with other components as possible.

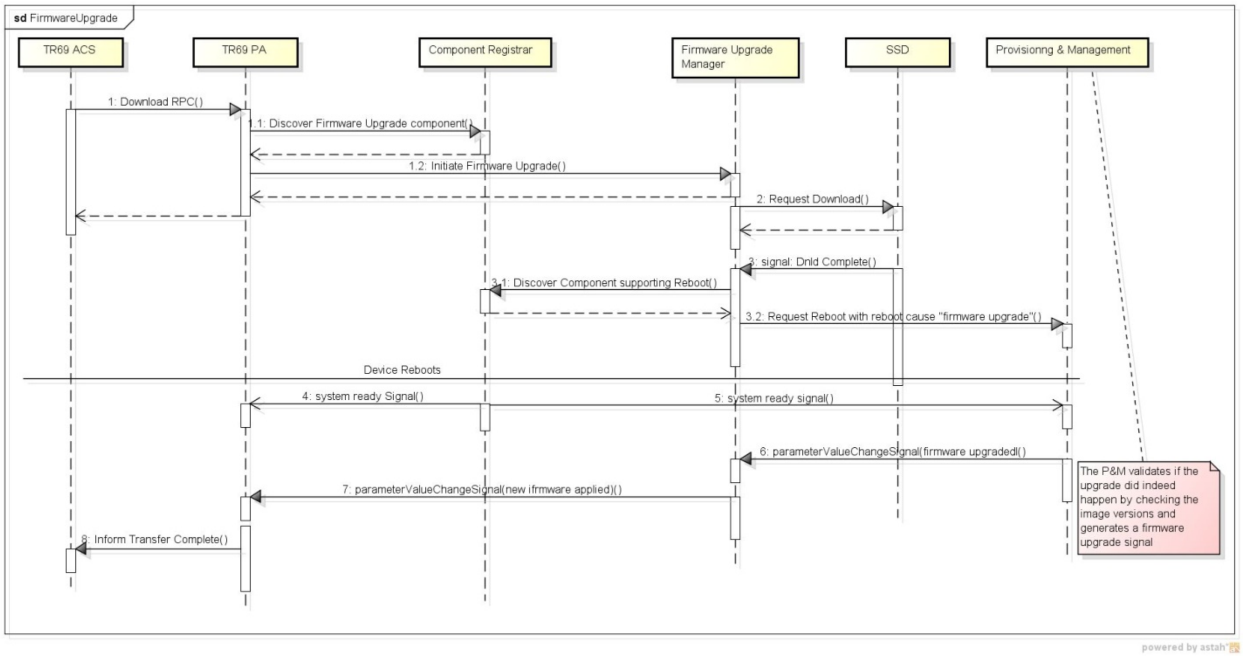
sd FirmwareUpgrade
TR69 ACS
TR69 PA
Component Registrar
Firmware Upgrade Manager
SSD
Provisionng & Management
1: Download RPC()
1.1: Discover Firmware Upgrade component()
1.2: Initiate Firmware Upgrade()
2: Request Download()
3: signal: Dnld Complete()
3.1: Discover Component supporting Reboot()
3.2: Request Reboot with reboot cause "firmware upgrade"()
Device Reboots
4: system ready Signal()
5: system ready signal()
6: parameterValueChangeSignal(firmware upgradedl()
7: parameterValueChangeSignal(new ifrmware applied)()
8: Inform Transfer Complete()
The P&M validates if the upgrade did indeed happen by checking the image versions and generates a firmware upgrade signal
powered by astah*

Figure 4 Firmware update data flow

Figure 5 demonstrates data flow of ChangeDUState RPC method call. The operation overall is similar to firmware upgrade process. Please refer Application Environment Manager Functional Component Specification for details.

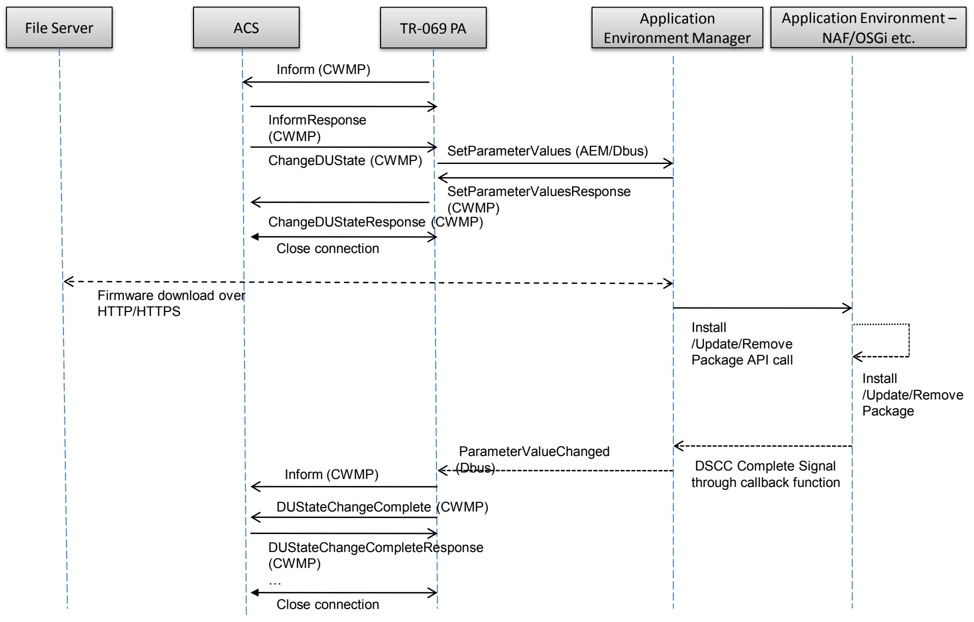
File Server
ACS
TR-069 PA
Application Environment Manager
Application Environment – NAF/OSGi etc.
Inform (CWMP)
InformResponse (CWMP)
ChangeDUState (CWMP)
SetParameterValues (AEM/Dbus)
SetParameterValuesResponse (CWMP)
ChangeDUStateResponse (CWMP)
Close connection
Firmware download over HTTP/HTTPS
Install /Update/Remove Package API call
Install /Update/Remove Package
ParameterValueChanged (Dbus)
DSCC Complete Signal through callback function
Inform (CWMP)
DUStateChangeComplete (CWMP)
DUStateChangeCompleteResponse (CWMP)
…
Close connection

Figure 5 TR-069 Package management operation data flow

Figure 6 demonstrates data flow of Vendor Configuration File download process as we have mentioned in section 5.2.2.

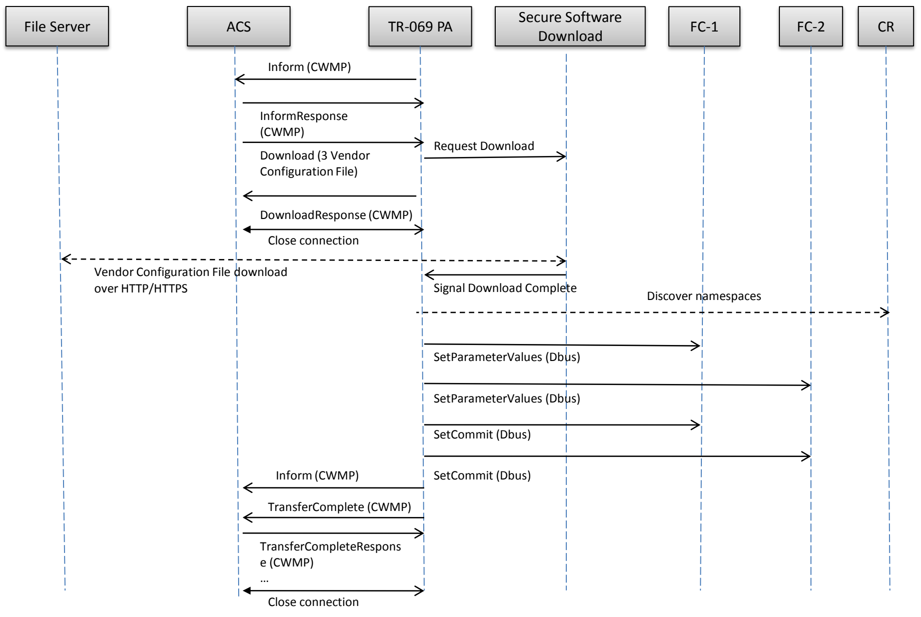

Figure 6 Vendor Configuration File Download

6 Porting Guide

The implementation of this component must be platform independent. This component relies on CCSP message bus to work properly. As long as message bus is portable, this component should be fine.

7 Glossary

The following list describes acronyms and definitions for terms used throughout this document:

- **PA**: Protocol Agent
- **CR**: Component Registry
- **PSM**: Persistent Storage Manager.
- **FC**: Functional Components

8 Attachments

TR-069 PA owns the following namespaces defined by TR-181. Please note some of parameters listed here may already be deprecated or not supported in first release.

8.1 TR-069 PA supported data model namespaces

Table 3 TR-069 PA supported data model

Device.DeviceInfo.SupportedDataModel.{i}.	object	-	This table contains details of the device's Current Supported Data Model. The table MUST describe the device's entire Supported Data Model. Therefore, if a device's Supported Data Model changes at run-time, entries will need to be added or removed as appropriate. Each table entry MUST refer to only a single Root Object or Service Object. The device MAY choose to use more than one table entry for a given Root Object or Service Object. Considering that every device has some form of a data model, this table MUST NOT be empty.	-	2.0

			At most one entry in this table can exist with a given value for *URL*.		
URL	string(256)	-	URL ([RFC3986]) that describes some or all of the device's Current Supported Data Model. The URL MUST reference an XML file which describes the appropriate part of the Supported Data Model. The referenced XML file MUST be compliant with the DT (Device Type) Schema that is described in Annex B/[TR-106a3], including any additional normative requirements referenced within the Schema. The XML file referenced by this URL MUST NOT change while the CPE is running, and SHOULD NOT change across a CPE reboot. Note that, if the same XML file is to be used for multiple CPE, this strongly suggests that the XML file referenced by this URL should *never* change. The URL MAY permit the XML file to be accessed at run-time, in which case, the XML file MAY be located within the CPE. Behavior in the event of an invalid URL, failure to access the referenced XML file, or an invalid XML file, is implementation-dependent.	-	2.0
URN	string(256)	-	URN ([RFC3986]) that is the value of the spec attribute in the DM (data model) Instance that defines the Root Object or Service Object referenced by this table entry. For example, if this table entry references a DT Instance that refers to the *Device:1.3* Root Object, the value of this parameter would be *urn:broadband-forum-org:tr-157-1-0-0*, because TR-157 defines *Device:1.3*. If the DT Instance instead referred to a vendor-specific Root Object, e.g. *X_EXAMPLE_Device:1.0* (derived from *Device:1.3*), the value of this parameter would be something like *urn:example-com:device-1-0-0*.	-	2.0
Features	string	-	Comma-separated list of strings. This parameter MUST list exactly the features that are defined using the top-level *feature* element in the DT Instance referenced by *URL*. For example, if the DT instance specified the following:	-	2.0

			<feature name="DNSServer"/> <feature name="Router"/> <feature name="X_MyDeviceFeature"/> then the value of this parameter might be *DNSServer,Router,X_MyDeviceFeature*. The order in which the features are listed is not significant.		
Device.ManagementServer.	object	-	This object contains parameters relating to the CPE's association with an ACS.	-	2.0
EnableCWMP	boolean	W	Enables and disables the CPE's support for CWMP. false means that CWMP support in the CPE is disabled, in which case the device MUST NOT send any Inform messages to the ACS or accept any Connection Request notifications from the ACS. true means that CWMP support on the CPE is enabled. The subscriber can re-enable the CPE's CWMP support either by performing a factory reset or by using a LAN-side protocol to change the value of this parameter back to true. The factory default value MUST be true.	-	2.0
URL	string(256)	W	URL, as defined in [RFC3986], for the CPE to connect to the ACS using the CPE WAN Management Protocol. This parameter MUST be in the form of a valid HTTP or HTTPS URL. The host portion of this URL is used by the CPE for validating the ACS certificate when using SSL or TLS. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset.	-	2.0
Username	string(256)	W	Username used to authenticate the CPE when making a connection to the ACS using the CPE WAN Management Protocol. This username is used only for HTTP-based authentication of the CPE. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset.	-	2.0
Password	string(256)	W	Password used to authenticate the CPE when making a connection to the ACS using the CPE WAN Management Protocol. This password is used only for HTTP-based authentication of the CPE. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS	-	2.0

			modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset. When read, this parameter returns an empty string, regardless of the actual value.		
PeriodicInformEnable	boolean	W	Whether or not the CPE MUST periodically send CPE information to the ACS using the Inform method call.	-	2.0
PeriodicInformInterval	unsignedInt-[1:]	W	The duration in seconds of the interval for which the CPE MUST attempt to connect with the ACS and call the Inform method if PeriodicInformEnable is true.	-	2.0
PeriodicInformTime	dateTime	W	An absolute time reference in UTC to determine when the CPE will initiate the periodic Inform method calls. Each Inform call MUST occur at this reference time plus or minus an integer multiple of the PeriodicInformInterval. PeriodicInformTime is used only to set the phase of the periodic Informs. The actual value of PeriodicInformTime can be arbitrarily far into the past or future. For example, if PeriodicInformInterval is 86400 (a day) and if PeriodicInformTime is set to UTC midnight on some day (in the past, present, or future) then periodic Informs will occur every day at UTC midnight. These MUST begin on the very next midnight, even if PeriodicInformTime refers to a day in the future. The Unknown Time value defined in Section 3.2/[TR-106a2] indicates that no particular time reference is specified. That is, the CPE MAY locally choose the time reference, and needs only to adhere to the specified PeriodicInformInterval. If absolute time is not available to the CPE, its periodic Inform behavior MUST be the same as if the PeriodicInformTime parameter was set to the Unknown Time value.	-	2.0
ParameterKey	string(32)	-	ParameterKey provides the ACS a reliable and extensible means to track changes made by the ACS. The value of ParameterKey MUST be equal to the value of the ParameterKey argument from the most recent successful SetParameterValues, AddObject, or DeleteObject method call from the ACS. The CPE MUST set ParameterKey to the value specified in the corresponding method arguments if and only if the method completes successfully and no fault response is generated. If a method call does not complete successfully (implying that the changes requested in the method did not take effect), the value of ParameterKey MUST NOT be modified. The CPE MUST only modify the value of ParameterKey as a result of	-	2.0

			SetParameterValues, AddObject, DeleteObject, or due to a factory reset. On factory reset, the value of ParameterKey MUST be set to an empty string.		
ConnectionRequestURL	string(256)	-	HTTP URL, as defined in [RFC3986], for an ACS to make a Connection Request notification to the CPE. In the form: http://host:port/path The host portion of the URL MAY be the IP address for the management interface of the CPE in lieu of a host name.	-	2.0
ConnectionRequestUsername	string(256)	W	Username used to authenticate an ACS making a Connection Request to the CPE.	-	2.0
ConnectionRequestPassword	string(256)	W	Password used to authenticate an ACS making a Connection Request to the CPE. When read, this parameter returns an empty string, regardless of the actual value.	-	2.0
UpgradesManaged	boolean	W	Indicates whether or not the ACS will manage upgrades for the CPE. If true, the CPE SHOULD NOT use other means other than the ACS to seek out available upgrades. If false, the CPE MAY use other means for this purpose. Note that an autonomous upgrade (reported via an "10 AUTONOMOUS TRANSFER COMPLETE" Inform Event code) SHOULD be regarded as a managed upgrade if it is performed according to ACS-specified policy.	-	2.0
KickURL	string(256)	-	Present only for a CPE that supports the Kicked RPC method. LAN-accessible URL, as defined in [RFC3986], from which the CPE can be kicked to initiate the Kicked RPC method call. MUST be an absolute URL including a host name or IP address as would be used on the LAN side of the CPE.	-	2.0
DownloadProgressURL	string(256)	-	Present only for a CPE that provides a LAN-side web page to show progress during a file download. LAN-accessible URL, as defined in [RFC3986], to which a web-server associated with the ACS MAY redirect a user's browser on initiation of a file download to observer the status of the download.	-	2.0
DefaultActiveNotificationThrottle	unsignedInt	W	This parameter is used to control throttling of active notifications sent by the CPE to the ACS. It defines the minimum number of seconds that the CPE MUST wait since the end of the last session with the ACS before establishing a new session for the purpose of delivering an active notification. In other words, if CPE needs to establish a new session with the ACS for the sole purpose of delivering an active notification, it MUST delay establishing	-	2.0

			such a session as needed to ensure that the minimum time since the last session completion has been met. The time is counted since the last successfully completed session, regardless of whether or not it was used for active notifications or other purposes. However, if connection to the ACS is established for purposes other than just delivering active notifications, including for the purpose of retrying a failed session, such connection MUST NOT be delayed based on this parameter value, and the pending active notifications MUST be communicated during that connection. The time of the last session completion does not need to be tracked across reboots.		
CWMPRetryMinimumWaitInterval	unsignedInt-[1:65535]	W	Configures the first session retry wait interval, in seconds, as specified in Section 3.2.1.1/[TR-069a2]. A value of 5 corresponds to the default behavior that is described in [TR-069a2]. The device MUST use a random value between CWMPRetryMinimumWaitInterval and (CWMPRetryMinimumWaitInterval * CWMPRetryIntervalMultiplier / 1000) as the first retry wait interval. Other values in the retry pattern MUST be calculated using this value as a starting point.	-	2.0
CWMPRetryIntervalMultiplier	unsignedInt-[1000:65535]	W	Configures the retry interval multiplier as specified in Section 3.2.1.1/[TR-069a2]. This value is expressed in units of 0.001. Hence the values of the multiplier range between 1.000 and 65.535. A value of 2000 corresponds to the default behavior that is described in [TR-069a2]. The device MUST use a random value between CWMPRetryMinimumWaitInterval and (CWMPRetryMinimumWaitInterval * CWMPRetryIntervalMultiplier / 1000) as the first retry wait interval. Other values in the retry pattern MUST be calculated using this value as a starting point.	-	2.0
Device.ManagementServer.AutonomousTransferCompletePolicy.	object	-	This object allows configuration of CPE policy for notification of AUTONOMOUS TRANSFER COMPLETE events, defined in [TR-069a2]. The CPE policy determines the conditions under which the CPE notifies the ACS of the completion of file transfers that were not specifically requested by the ACS.	-	2.0
Enable	boolean	W	Enable/disable CPE notification of AUTONOMOUS TRANSFER COMPLETE events to the ACS.	-	2.0
TransferTypeFilter	string	W	Indicates the transfer types that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER COMPLETE events. Transfer types not	-	2.0

			indicated by this filter MUST NOT be included when the CPE notifies the ACS. Enumeration of: • Upload • Download • Both (upload and Download) Note that this includes any backup or restore operations that were not specifically requested by the ACS. A backup is regarded as an Upload and a restore is regarded as a Download.		
ResultTypeFilter	string	W	Indicates the transfer results that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER COMPLETE events. Transfer results omitted from this list MUST NOT be included when the CPE notifies the ACS. Note that this includes any backup or restore operations that were not specifically requested by the ACS. A backup is regarded as an Upload and a restore is regarded as a Download. Enumeration of: • Success (the autonomous file transfer completed successfully; i.e., the FaultCode was zero) • Failure (the autonomous file transfer did not complete successfully; i.e., the FaultCode was non-zero) • Both (success and Failure)	-	2.0
FileTypeFilter	string(1024)	W	Comma-separated list (maximum length 1024) of strings. Indicates the file types that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER COMPLETE events. File types omitted from this list MUST NOT be included when the CPE notifies the ACS. • 1 Firmware Upgrade Image (Download Only) • 2 Web Content (Download Only) • 3 Vendor Configuration File (download or Upload) • 4 Vendor Log File (Upload Only) • X [0-9A-F]{6} .* (For Vendor-Specific File Types, could be for either Download or Upload) Additionally, the following format is defined to allow the unique definition of vendor-specific file types:	-	2.0

			• "X <OUI> <Vendor-specific identifier>" <OUI> is replaced by a 6 hexadecimal-digit OUI (organizationally unique identifier) as defined in [OUI], with all upper-case letters and any leading zeros included. The OUI used for a given vendor-specific file type MUST be one that is assigned to the organization that defined this file type (which is not necessarily the same as the vendor of the CPE or ACS). Note that an empty string indicates that all file types are excluded from this filter, effectively disabling CPE notification of AUTONOMOUS TRANSFER COMPLETE events to the ACS.		
Device.ManagementServer.DUStateChangeComplPolicy.	object	-	This object allows configuration of CPE policy for notification of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events defined in [TR-069a3]. The CPE policy determines the conditions under which the CPE notifies the ACS of the completion of Deployment Unit state changes that were not specifically requested via CWMP.	-	1.7
Enable	boolean	W	Enables/Disables CPE notification of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events to the ACS.	-	1.7
OperationTypeFilter	string	W	Comma-separated list of strings. Indicates the Deployment Unit operations that MUST be included when the CPE notifies the ACS of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events. Operation types not indicated by this list MUST NOT be included when the CPE notifies the ACS. An empty string is essentially the same as setting Enable to false. Each list item is an enumeration of: • Install • Update • Uninstall	-	1.7
ResultTypeFilter	string	W	Indicates the Deployment Unit state change results that MUST be included when the CPE notifies the ACS of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events. State change results omitted from this filter MUST NOT be included when the CPE notifies the ACS. Enumeration of: • Success (The autonomous state change completed successfully; i.e., the FaultCode was zero)	-	1.7

			• Failure (The autonomous state change did not complete successfully; i.e., the FaultCode was non-zero) • Both (All result types independent of Success or Failure)		
FaultCodeFilter	string	W	Comma-separated list of strings. Indicates the Deployment Unit state change fault codes that MUST be included when the CPE notifies the ACS of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events. State change fault codes omitted from this filter MUST NOT be included when the CPE notifies the ACS. This filter has no effect on the notification of a successful autonomous state change. This filter only applies when ResultTypeFilter is set to either Failure or Both. An empty string means that failed autonomous state changes will not be sent to the ACS. Each list item is an enumeration of: • 9001 • 9003 • 9012 • 9013 • 9015 • 9016 • 9017 • 9018 • 9022 • 9023 • 9024 • 9025 • 9026 • 9027 • 9028 • 9029 • 9030 • 9031 • 9032	-	1.7

8.2 TR-069 PA XML Data Model

Table 4 Object “Device.DeviceInfo.SupportedDataModel.{i}.”

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<object name="Device.DeviceInfo.SupportedDataModel.{i}." access="readOnly" minEntries="0" maxEntries="unbounded"
numEntriesParameter="SupportedDataModelNumberOfEntries">
  <description>
    This table contains details of the device's Current Supported Data Model.
    The table MUST describe the device's entire Supported Data Model.  Therefore, if a device's Supported Data
    Model changes at run-time, entries will need to be added or removed as appropriate.
    Each table entry MUST refer to only a single Root Object or Service Object.  The device MAY choose to use more
   than one table entry for a given Root Object or Service Object.
    Considering that every device has some form of a data model, this table MUST NOT be empty.
  </description>
  <uniqueKey>
    <parameter ref="URL"/>
  </uniqueKey>


  <parameter name="URL" access="readOnly">
    <description>
      URL ({{bibref|RFC3986}}) that describes some or all of the device's Current Supported Data Model.
      The URL MUST reference an XML file which describes the appropriate part of the Supported Data Model.
      The referenced XML file MUST be compliant with the DT (Device Type) Schema that is described in
     {{bibref|TR-106a3|Annex B}}, including any additional normative requirements referenced within the Schema.
      The XML file referenced by this URL MUST NOT change while the CPE is running, and SHOULD NOT
     change across a CPE reboot.  Note that, if the same XML file is to be used for multiple CPE, this strongly
      suggests that the XML file referenced by this URL should "never" change.
      The URL MAY permit the XML file to be accessed at run-time, in which case, the XML file MAY be located
      within the CPE.
      Behavior in the event of an invalid URL, failure to access the referenced XML file, or an invalid XML file, is
      implementation-dependent.</description>
    <syntax><string><size maxLength="256"/></string></syntax>
  </parameter>
  <parameter name="URN" access="readOnly">
    <description>
      URN ({{bibref|RFC3986}}) that is the value of the spec attribute in the DM (data model) Instance that defines
      the Root Object or Service Object referenced by this table entry.
      For example, if this table entry references a DT Instance that refers to the "Device:1.3" Root Object, the value
      of this parameter would be "urn:broadband-forum-org:tr-157-1-0-0", because TR-157 defines "Device:1.3".  If
      the DT Instance instead referred to a vendor-specific Root Object, e.g. "X_EXAMPLE_Device:1.0" (derived
      from "Device:1.3"), the value of this parameter would be something like "urn:example-com:device-1-0-0".
    </description>
    <syntax><string><size maxLength="256"/></string></syntax>
  </parameter>
  <parameter name="Features" access="readOnly">
    <description>
      This parameter MUST list exactly the features that are defined using the top-level "feature" element in the DT
      Instance referenced by {{param|URL}}.
      For example, if the DT instance specified the following:
      :&lt;feature name="DNSServer"/&gt;
      :&lt;feature name="Router"/&gt;
      :&lt;feature name="X_MyDeviceFeature"/&gt;
      then the value of this parameter might be "DNSServer,Router,X_MyDeviceFeature".  The order in which the
      features are listed is not significant.
    </description>
    <syntax><list/><string/></syntax>
  </parameter>
</object>
```

Table 5 Object "Device.ManagementServer."

```
<object name="Device.ManagementServer." access="readOnly" minEntries="1" maxEntries="1">
  <description>This object contains parameters relating to the CPE's association with an ACS.</description>
  <parameter name="EnableCWMP" access="readWrite">
    <description>Enables and disables the CPE's support for CWMP.
      {{false}} means that CWMP support in the CPE is disabled, in which case the device MUST NOT send any Inform
      messages to the ACS or  accept any Connection Request notifications from the ACS.
      {{true}} means that CWMP support on the CPE is enabled.
      The subscriber can re-enable the CPE's CWMP support either by performing a factory reset or by using a LAN-side
      protocol to change  the value of this parameter back to {{true}}.</description>
    <syntax>
      <boolean/>
      <default type="factory" value="true"/>
    </syntax>
  </parameter>
  <parameter name="URL" access="readWrite">
    <description>URL, as defined in {{bibref|RFC3986}}, for the CPE to connect to the ACS using the CPE WAN
      Management Protocol.
      This parameter MUST be in the form of a valid HTTP or HTTPS URL.
      The "host" portion of this URL is used by the CPE for validating the ACS certificate when using SSL or TLS.
      Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value.  If an ACS
      modifies the  value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is
      restored as the result of a  factory reset.</description>
    <syntax>
      <string>
        <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="Username" access="readWrite">
    <description>Username used to authenticate the CPE when making a connection to the ACS using the CPE WAN
      Management Protocol.
      This username is used only for HTTP-based authentication of the CPE.
      Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value.  If an ACS
      modifies the  value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is
      restored as the result of a  factory reset.</description>
    <syntax>
      <string>
        <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="Password" access="readWrite">
    <description>Password used to authenticate the CPE when making a connection to the ACS using the CPE WAN
      Management Protocol.
      This password is used only for HTTP-based authentication of the CPE.
      Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value.  If an ACS
      modifies the  value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is
      restored as the result of a  factory reset.</description>
    <syntax hidden="true">
      <string>
```

```
        <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="PeriodicInformEnable" access="readWrite">
    <description>Whether or not the CPE MUST periodically send CPE information to the ACS using the Inform method
      call.</description>
    <syntax>
      <boolean/>
    </syntax>
  </parameter>
  <parameter name="PeriodicInformInterval" access="readWrite">
    <description>The duration in {{units}} of the interval for which the CPE MUST attempt to connect with the ACS and
      call the  Inform method if {{param|PeriodicInformEnable}} is {{true}}.</description>
    <syntax>
      <unsignedInt>
        <range minInclusive="1"/>
        <units value="seconds"/>
      </unsignedInt>
    </syntax>
  </parameter>
  <parameter name="PeriodicInformTime" access="readWrite">
    <description>An absolute time reference in UTC to determine when the CPE will initiate the periodic Inform method
      calls.   Each Inform call MUST occur at this reference time plus or minus an integer multiple of the
      {{param|PeriodicInformInterval}}.
      {{param}} is used only to set the "phase" of the periodic Informs.  The actual value of {{param}} can be arbitrarily far
      into the  past or future.
      For example, if {{param|PeriodicInformInterval}} is 86400 (a day) and if {{param}} is set to UTC midnight on some
      day (in the past,  present, or future) then periodic Informs will occur every day at UTC midnight.  These MUST begin
      on the very next midnight, even if  {{param}} refers to a day in the future.
      The Unknown Time value defined in {{bibref|TR-106a2|section 3.2}} indicates that no particular time reference is
      specified.  That is,  the CPE MAY locally choose the time reference, and needs only to adhere to the specified
      {{param|PeriodicInformInterval}}.
      If absolute time is not available to the CPE, its periodic Inform behavior MUST be the same as if the {{param}}
      parameter was set to  the Unknown Time value.</description>
    <syntax>
      <dateTime/>
    </syntax>
  </parameter>
  <parameter name="ParameterKey" access="readOnly" activeNotify="canDeny" forcedInform="true">
    <description>{{param}} provides the ACS a reliable and extensible means to track changes made by the ACS.  The
      value of  {{param}} MUST be equal to the value of the ParameterKey argument from the most recent successful
      SetParameterValues, AddObject, or  DeleteObject method call from the ACS.
      The CPE MUST set {{param}} to the value specified in the corresponding method arguments if and only if the method
      completes  successfully and no fault response is generated.  If a method call does not complete successfully (implying
      that the changes  requested in the method did not take effect), the value of {{param}} MUST NOT be modified.
      The CPE MUST only modify the value of {{param}} as a result of SetParameterValues, AddObject, DeleteObject, or
      due to a factory  reset.  On factory reset, the value of {{param}} MUST be set to {{empty}}.</description>
    <syntax>
      <string>
        <size maxLength="32"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="ConnectionRequestURL" access="readOnly" forcedInform="true"
      activeNotify="forceDefaultEnabled">
    <description>HTTP URL, as defined in {{bibref|RFC3986}}, for an ACS to make a Connection Request notification to
      the CPE.
      In the form:
        : http://host:port/path
      The "host" portion of the URL MAY be the IP address for the management interface of the CPE in lieu of a host
      name.</description>
    <syntax>
      <string>
      <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="ConnectionRequestUsername" access="readWrite">
    <description>Username used to authenticate an ACS making a Connection Request to the CPE.</description>
    <syntax>
      <string>
      <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="ConnectionRequestPassword" access="readWrite">
    <description>Password used to authenticate an ACS making a Connection Request to the CPE.</description>
    <syntax hidden="true">
      <string>
      <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="UpgradesManaged" access="readWrite">
    <description>Indicates whether or not the ACS will manage upgrades for the CPE.  If {{true}}, the CPE SHOULD NOT
      use other  means other than the ACS to seek out available upgrades.  If {{false}}, the CPE MAY use other means for
      this purpose.
      Note that an autonomous upgrade (reported via an "10 AUTONOMOUS TRANSFER COMPLETE" Inform Event
      code) SHOULD be regarded as a managed  upgrade if it is performed according to ACS-specified
      policy.</description>
    <syntax>
      <boolean/>
    </syntax>
  </parameter>
  <parameter name="KickURL" access="readOnly">
    <description>Present only for a CPE that supports the Kicked RPC method.
      LAN-accessible URL, as defined in {{bibref|RFC3986}}, from which the CPE can be "kicked" to initiate the Kicked
      RPC method call.   MUST be an absolute URL including a host name or IP address as would be used on the LAN side
      of the CPE.</description>
    <syntax>
      <string>
      <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="DownloadProgressURL" access="readOnly">
    <description>Present only for a CPE that provides a LAN-side web page to show progress during a file download.
      LAN-accessible URL, as defined in {{bibref|RFC3986}}, to which a web-server associated with the ACS MAY
      redirect a user's browser on  initiation of a file download to observer the status of the download.</description>
    <syntax>
```

```
      <string>
        <size maxLength="256"/>
      </string>
    </syntax>
  </parameter>
  <parameter name="DefaultActiveNotificationThrottle" access="readWrite">
    <description>This parameter is used to control throttling of active notifications sent by the CPE to the ACS.  It defines
        the  minimum number of {{units}} that the CPE MUST wait since the end of the last session with the ACS before
        establishing a new session for  the purpose of delivering an active notification.
        In other words, if CPE needs to establish a new session with the ACS for the sole purpose of delivering an active
        notification, it  MUST delay establishing such a session as needed to ensure that the minimum time since the last
        session completion has been met.
       The time is counted since the last successfully completed session, regardless of whether or not it was used for active
       notifications  or other purposes. However, if connection to the ACS is established for purposes other than just
       delivering active notifications,  including for the purpose of retrying a failed session, such connection MUST NOT be
       delayed based on this parameter value, and the  pending active notifications MUST be communicated during that
       connection.
       The time of the last session completion does not need to be tracked across reboots.</description>
    <syntax>
      <unsignedInt>
        <units value="seconds"/>
      </unsignedInt>
    </syntax>
  </parameter>
  <parameter name="CWMPRetryMinimumWaitInterval" access="readWrite">
    <description>Configures the first session retry wait interval, in {{units}}, as specified in {{bibref|TR-069a2|section
        3.2.1.1}}.
        A value of 5 corresponds to the default behavior that is described in {{bibref|TR-069a2}}.
        The device MUST use a random value between {{param}}  and ({{param}} *
        {{param|CWMPRetryIntervalMultiplier}} / 1000) as the first  retry wait interval.  Other values in the retry pattern
        MUST be calculated using this value as a starting point.</description>
    <syntax>
      <unsignedInt>
        <range minInclusive="1" maxInclusive="65535"/>
        <units value="seconds"/>
      </unsignedInt>
    </syntax>
  </parameter>
  <parameter name="CWMPRetryIntervalMultiplier" access="readWrite">
    <description>Configures the retry interval multiplier as specified in {{bibref|TR-069a2|section 3.2.1.1}}.
        This value is expressed in units of 0.001.  Hence the values of the multiplier range between 1.000 and 65.535.
        A value of 2000 corresponds to the default behavior that is described in {{bibref|TR-069a2}}.
        The device MUST use a random value between {{param|CWMPRetryMinimumWaitInterval}}  and
        ({{param|CWMPRetryMinimumWaitInterval}} *  {{param}} / 1000) as the first retry wait interval.  Other values in
        the retry pattern MUST be calculated using this value as a  starting point.</description>
    <syntax>
      <unsignedInt>
        <range minInclusive="1000" maxInclusive="65535"/>
      </unsignedInt>
    </syntax>
  </parameter>
  <parameter name="UDPConnectionRequestAddress" access="readOnly">
    <description>Address and port to which an ACS MAY send a UDP Connection Request to the CPE (see {{bibref|TR-
        069a2|Annex G}}).
        This parameter is represented in the form of an Authority element as defined in {{bibref|RFC3986}}.  The value
        MUST be in one of the  following two forms:
```

```
                    : host:port
                    : host
                   * When {{param|STUNEnable}} is {{true}}, the "host" and "port" portions of this parameter MUST represent the
                     public address and port corresponding to the NAT binding through which the ACS can send UDP Connection
                     Request messages (once this information is learned by the CPE through the use of STUN).
                  * When {{param|STUNEnable}} is {{false}}, the "host" and "port" portions of the URL MUST represent the local IP
                    address and port on which the CPE is listening for UDP Connection Request messages.
                The second form of this parameter MAY be used only if the port value is equal to "80".</description>
              <syntax>
                <string>
                  <size maxLength="256"/>
                </string>
              </syntax>
            </parameter>
            <parameter name="STUNEnable" access="readWrite">
              <description>Enables or disables the use of STUN by the CPE.  This applies only to the use of STUN in association with
                    the  ACS to allow UDP Connection Requests.</description>
              <syntax>
                <boolean/>
              </syntax>
            </parameter>
            <parameter name="STUNServerAddress" access="readWrite">
              <description>Host name or IP address of the STUN server for the CPE to send Binding Requests if STUN is enabled via
                    {{param|STUNEnable}}.
                    If is {{empty}} and {{param|STUNEnable}} is {{true}}, the CPE MUST use the address of the ACS extracted from
                    the host portion of the  ACS URL.</description>
              <syntax>
                <string>
                  <size maxLength="256"/>
                </string>
              </syntax>
            </parameter>
            <parameter name="STUNServerPort" access="readWrite">
              <description>Port number of the STUN server for the CPE to send Binding Requests if STUN is enabled via
                    {{param|STUNEnable}}.
                    By default, this SHOULD be the equal to the default STUN port, 3478.</description>
              <syntax>
                <unsignedInt>
                  <range minInclusive="0" maxInclusive="65535"/>
                </unsignedInt>
              </syntax>
            </parameter>
            <parameter name="STUNUsername" access="readWrite">
              <description>If is not {{empty}}, the value of the STUN USERNAME attribute to be used in Binding Requests (only if
                    message  integrity has been requested by the STUN server).
                    If is {{empty}}, the CPE MUST NOT send STUN Binding Requests with message integrity.</description>
              <syntax>
                <string>
                  <size maxLength="256"/>
                </string>
              </syntax>
            </parameter>
            <parameter name="STUNPassword" access="readWrite">
              <description>The value of the STUN Password to be used in computing the MESSAGE-INTEGRITY attribute to be used
                    in Binding  Requests (only if message integrity has been requested by the STUN server).</description>
```

```
      <syntax hidden="true">
        <string>
          <size maxLength="256"/>
        </string>
      </syntax>
    </parameter>
    <parameter name="STUNMaximumKeepAlivePeriod" access="readWrite">
      <description>If STUN Is enabled, the maximum period, in {{units}}, that STUN Binding Requests MUST be sent by the
           CPE for the  purpose of maintaining the binding in the Gateway.  This applies specifically to Binding Requests sent
           from the UDP Connection  Request address and port.
           A value of -1 indicates that no maximum period is specified.</description>
      <syntax>
        <int>
          <range minInclusive="-1"/>
          <units value="seconds"/>
        </int>
      </syntax>
    </parameter>
    <parameter name="STUNMinimumKeepAlivePeriod" access="readWrite">
      <description>If STUN Is enabled, the minimum period, in {{units}}, that STUN Binding Requests can be sent by the
            CPE for the  purpose of maintaining the binding in the Gateway.  This limit applies only to Binding Requests sent
            from the UDP Connection Request  address and port, and only those that do not contain the BINDING-CHANGE
            attribute.  This limit does not apply to retransmissions  following the procedures defined in
           {{bibref|RFC3489}}.</description>
      <syntax>
        <unsignedInt>
          <units value="seconds"/>
        </unsignedInt>
      </syntax>
    </parameter>
    <parameter name="NATDetected" access="readOnly">
      <description>When STUN is enabled, this parameter indicates whether or not the CPE has detected address and/or port
            mapping  in use.
            A {{true}} value indicates that the received MAPPED-ADDRESS in the most recent Binding Response differs from
           the CPE's source address and port.
           When {{param|STUNEnable}} is {{false}}, this value MUST be {{false}}.</description>
      <syntax>
        <boolean/>
      </syntax>
    </parameter>
    <parameter name="ManageableDeviceNumberOfEntries" access="readOnly">
      <description>{{numentries}}</description>
      <syntax>
        <unsignedInt/>
      </syntax>
    </parameter>
  </object>
```

Table 6 Object "Device.ManagementServer.AutonomousTransferCompletePolicy."

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<object name="Device.ManagementServer.AutonomousTransferCompletePolicy." access="readOnly" minEntries="1"
maxEntries="1">
  <description>
    This object allows configuration of CPE policy for notification of AUTONOMOUS TRANSFER COMPLETE events,
    defined in {{bibref|TR-069a2}}.
    The CPE policy determines the conditions under which the CPE notifies the ACS of the completion of file transfers that
    were not specifically requested by the ACS.
  </description>


  <parameter name="Enable" access="readWrite">
    <description>Enable/disable CPE notification of AUTONOMOUS TRANSFER COMPLETE events to the
      ACS.</description>
    <syntax><boolean/></syntax>
  </parameter>
  <parameter name="TransferTypeFilter" access="readWrite">
    <description>
      Indicates the transfer types that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER
      COMPLETE events.   Transfer types not indicated by this filter MUST NOT be included when the CPE notifies the
      ACS. {{enum}}
      Note that this includes any backup or restore operations that were not specifically requested by the ACS.  A backup is
      regarded as an Upload and a restore is regarded as a Download.
    </description>
    <syntax>
      <string>
        <enumeration value="Upload"/>
        <enumeration value="Download"/>
        <enumeration value="Both">
          <description>Upload and Download</description>
        </enumeration>
      </string>
    </syntax>
  </parameter>
  <parameter name="FileTypeFilter" access="readWrite">
    <description>
      {{list}} Indicates the file types that MUST be included when the CPE notifies the ACS of AUTONOMOUS
      TRANSFER COMPLETE events.  File types omitted from this list MUST NOT be included when the CPE notifies the
      ACS.
      {{pattern}}
      Additionally, the following format is defined to allow the unique definition of vendor-specific file types:
      * ""X &lt;OUI&gt; &lt;Vendor-specific identifier&gt;""
      &lt;OUI&gt; is replaced by a 6 hexadecimal-digit OUI (organizationally unique identifier) as defined in
      {{bibref|OUI}}, with all upper-case letters and any leading zeros included. The OUI used for a given vendor-specific
      file type MUST be one that is assigned to the organization that defined this file type (which is not necessarily the same
      as the vendor of the CPE or ACS).
      Note that {{empty}} indicates that all file types are excluded from this filter, effectively disabling CPE notification of
      AUTONOMOUS TRANSFER COMPLETE events to the ACS.
    </description>
    <syntax>
      <list><size maxLength="1024"/></list>
      <string>
        <pattern value="1 Firmware Upgrade Image">
          <description>Download Only</description>
        </pattern>
        <pattern value="2 Web Content">
          <description>Download Only</description>
        </pattern>
```

```
                <pattern value="3 Vendor Configuration File">
                    <description>Download or Upload</description>
                </pattern>
                <pattern value="4 Vendor Log File">
                    <description>Upload Only</description>
                </pattern>
                <pattern value="X [0-9A-F]{6} .*">
                    <description>For Vendor-Specific File Types, could be for either Download or Upload</description>
                </pattern>
            </string>
        </syntax>
    </parameter>
</object>
```

Table 7 Object "Device.ManagementServer.AutonomousTransferCompletePolicy."

```
<?xml version="1.0" encoding="UTF-8"?>


<object name="Device.ManagementServer.DUStateChangeComplPolicy." access="readOnly" minEntries="1"
maxEntries="1">
    <description>
        This object allows configuration of CPE policy for notification of "12 AUTONOMOUS DU STATE CHANGE
        COMPLETE" events defined in {{bibref|TR-069a3}}.
        The CPE policy determines the conditions under which the CPE notifies the ACS of the completion of Deployment Unit
        state changes that were not specifically requested via CWMP.
    </description>


    <parameter name="Enable" access="readWrite">
        <description>Enables/Disables CPE notification of "12 AUTONOMOUS DU STATE CHANGE COMPLETE" events to
            the ACS.</description>
        <syntax><boolean/></syntax>
    </parameter>
    <parameter name="OperationTypeFilter" access="readWrite">
        <description>
            Indicates the Deployment Unit operations that MUST be included when the CPE notifies the ACS of "12
            AUTONOMOUS DU STATE CHANGE COMPLETE" events.
            Operation types not indicated by this list MUST NOT be included when the CPE notifies the ACS.  {{empty}} is
            essentially the same as setting {{param|Enable}} to {{false}}.
        </description>
        <syntax>
            <list/>
            <string>
                <enumeration value="Install"/>
                <enumeration value="Update"/>
                <enumeration value="Uninstall"/>
            </string>
        </syntax>
    </parameter>
    <parameter name="ResultTypeFilter" access="readWrite">
        <description>
            Indicates the Deployment Unit state change results that MUST be included when the CPE notifies the ACS of "12
            AUTONOMOUS DU STATE CHANGE COMPLETE" events.  State change results omitted from this filter MUST
            NOT be included when the CPE notifies the ACS.
        </description>
        <syntax>
            <string>
                <enumeration value="Success">
                  <description>The autonomous state change completed successfully; i.e., the FaultCode was zero</description>
                </enumeration>
                <enumeration value="Failure">
                  <description>The autonomous state change did not complete successfully; i.e., the FaultCode was non-
                    zero</description>
                </enumeration>
                <enumeration value="Both">
                  <description>All result types independent of Success or Failure</description>
                </enumeration>
            </string>
        </syntax>
    </parameter>
    <parameter name="FaultCodeFilter" access="readWrite">
        <description>
            Indicates the Deployment Unit state change fault codes that MUST be included when the CPE notifies the ACS of "12
            AUTONOMOUS DU STATE CHANGE COMPLETE" events.  State change fault codes omitted from this filter
            MUST NOT be included when the CPE notifies the ACS.
            This filter has no effect on the notification of a successful autonomous state change. This filter only applies when
             {{param|ResultTypeFilter}} is set to either {{enum|Failure|ResultTypeFilter}} or {{enum|Both|ResultTypeFilter}}.
             {{empty}} means that failed autonomous state changes will not be sent to the ACS.
        </description>
        <syntax>
            <list/>
            <string>
                <enumeration value="9001"/>
                <enumeration value="9003"/>
                <enumeration value="9012"/>
                <enumeration value="9013"/>
                <enumeration value="9015"/>
                <enumeration value="9016"/>
                <enumeration value="9017"/>
                <enumeration value="9018"/>
                <enumeration value="9022"/>
                <enumeration value="9023"/>
                <enumeration value="9024"/>
                <enumeration value="9025"/>
                <enumeration value="9026"/>
                <enumeration value="9027"/>
                <enumeration value="9028"/>
                <enumeration value="9029"/>
                <enumeration value="9030"/>
                <enumeration value="9031"/>
                <enumeration value="9032"/>
            </string>
        </syntax>
    </parameter>
</object>
```

End of Document