



CHAPTER 2

Extensible Markup Language Processing

Revised: August 10, 2011, OL-25014-01

This chapter describes the Extensible Markup Language (XML) process in the Simple Object Access Protocol (SOAP) interface.

Basic XML and SOAP Components

Along with XML, the primary component of the SOAP interface is the Tomcat web application. In addition, there are dependencies on related components in the managed object (MO) Java package. The required packages are as follows:

- Apache XML
- Xerces (parser)
- ECS (XML Document Building Tool Kit)

These packages interoperate with the central to the functional components in the Cisco BTS 10200 Softswitch. There is no larger object model applied to the system. The SOAP interface supports a variety of packet telephony applications.

XML in the SOAP Interface

This section describes how to use XML in the SOAP interface. Terms used in this section follow those used in XML specifications. This is to avoid confusion in the use of terms such as *element*, *subelement*, and *attribute*.

CIS Functions

This section provides the schemas that perform client-side verification of the XML document structure. These schemas cover the following items:

- ManagedObject
- Request
- Reply

The schema for a *ManagedObject* follows the format listed below.

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

  <xs:element name="ManagedObject">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="MOAttribute" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Verb" type="xs:string" use="required"/>
      <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="MOAttribute">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Required"/>
        <xs:element ref="Type"/>
        <xs:element ref="Default"/>
        <xs:element ref="Width"/>
        <xs:element ref="HelpText"/>
        <xs:element ref="Label"/>
        <xs:element ref="Parser" minOccurs="0"/>
        <xs:element ref="Permitted" minOccurs="0"/>
        <xs:element ref="Fk" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Required" type="xs:boolean"/>

  <xs:element name="Type">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="single"/>
        <xs:enumeration value="text"/>
        <xs:enumeration value="multi"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="Default" type="xs:string"/>

  <xs:element name="HelpText" type="xs:string"/>

  <xs:element name="Label" type="xs:string"/>

  <xs:element name="Noun" type="xs:string"/>

  <xs:element name="Param" type="xs:string"/>

  <xs:element name="Parser">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="JavaScript"/>
        <xs:element ref="RegExp"/>
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="JavaScript" type="xs:string"/>

```

```

<xs:element name="RegExp" type="xs:string"/>

<xs:element name="Permitted" type="xs:string"/>

<xs:element name="Width" type="xs:int" />

<xs:element name="Fk">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Noun"/>
      <xs:element ref="Param"/>
      <xs:element ref="Fk" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>

```

The schema for a *Request* follows the format listed below.

```

<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Request">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Entry" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Verb" type="xs:string" use="required"/>
        <xs:attribute name="Noun" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="Entry">
      <xs:complexType>
        <xs:attribute name="Key" type="xs:string" use="required"/>
        <xs:attribute name="Value" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>

```

The schema for a *Reply* follows the format listed below.

```

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

  <xs:element name="Reply">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Status"/>
        <xs:element ref="Reason"/>
        <xs:element ref="Size"/>
        <xs:element ref="AbsoluteSize"/>
        <xs:element ref="StartRow"/>
        <xs:element ref="DataTable"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Status" type="xs:boolean"/>

  <xs:element name="Reason" type="xs:string"/>

  <xs:element name="Size" type="xs:integer"/>

```

```

<xs:element name="AbsoluteSize" type="xs:integer" />

<xs:element name="StartRow" type="xs:integer" />

<xs:element name="DataTable">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Row" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Row">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Column" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:integer" />
  </xs:complexType>
</xs:element>

<xs:element name="Column" >
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="id" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</xs:schema>

```

The interface definition language (IDL) allows access to XML description documents for each noun and verb combination. For example, the **add subscriber** command generates a matching XML document that defines the element and attributes of this command. The IDL allows command processing based on a well-formed but unverified XML document.

The IDL allows command access to supported media gateway (MGW) devices. The IDL commands do not follow the XML access format defined in the schema. The CIS supports MGW commands that are native to the MGW internal command structure.

XML documents that originate in the BTS 10200 are dynamically generated and include all command-description documents.

ManagedObject

A *ManagedObject* has one element, the *MOAttribute*. A *ManagedObject* also has two attributes: the *id* of the *ManagedObject* and the *verb*. The *id* represents the object on which some action is to be taken. The *verb* indicates the action to be taken. For example, subscriber, or termination is a valid ID. This *id* a required attribute.

The following list describes the various parts of the schema and the valid values for each part:

- **Verb**—This attribute defines the action to take on a given ManagedObject. This is a required attribute and is composed of character data.
- **MOAttribute**—The ManagedObject can contain none, one, or more of these elements. It has one attribute named *id*. This character data acts as a label for the element. The order of these elements does not imply any specific behavior. They are arbitrarily listed.
- **Required**—This subelement has two values defined as true or false.
- **Type**—This subelement defines whether the MOAttribute has a single value, multiple values, or is a text. The multi or single option implies that the permitted element offers a list of choices.
- **Default**—This subelement indicates the default value for the MOAttribute.
- **Width**—This subelement indicates the total field width of the data. For example, if the MOAttribute is a description, this indicates the length of the description.
- **HelpText**—This subelement offers a brief text to indicate the nature of the MOAttribute.
- **Permitted**—This subelement specifies the possible values or ranges for the MOAttribute.
- **Parser**—This subelement indicates what type of validation is required. There is a single attribute to this subelement. The attribute is specified as character data in an ID field. The subelements are listed below.
 - **JavaScript**—This subelement represents a JavaScript, which performs validation or regular expression matching.
 - **RegExp**—This subelement defines the regular expression in character data format.

Request Schema

The Request schema consists of a single element that contains no subelements or one or more Entry subelements. An Entry subelement can have as many as two attributes.

The following list presents the parts of the Request schema and the values that are valid for those parts:

- **Noun**—This attribute defines the item on which some operation is requested. This is expressed as character data.
- **Verb**—This attribute defines the action to perform on the “Noun” attribute. This is expressed as character data.

The *Entry* element is allowed to be empty. It can also contain two attributes. These attributes are defined as follows:

- **Key**—This is the *id* value derived from the MOAttribute in the ManagedObject. It is expressed as character data.
- **Value**—This is the client-derived value assigned to the Key attribute. The Value is expressed as character data. It should also conform to the subelements in MOAttribute from which this Key/Value pair was derived.

Reply Schema

The Reply schema defines the structure of returned data generated in response to a Request. The Reply contains three elements and no attributes. These elements are defined as follows:

- **Status**—The Status element has two possible values. Either true or false is applied to this element.

- **Reason**—The Reason element contains character data. This element explains the cause for an error in processing a command or returns a success indication.
- **DataTable**—This element has one attribute and one subelement, which are defined below. The DataTable is used as the container for data that results from the execution of a request. Each Reply can contain a *DataTable* element.
 - **Row**—This subelement defines a single complete item of data. A *DataTable* can contain one or more Row subelements. A Row has one attribute. This character data defines the row ID. The ID is always a sequential value based on the number of returned rows. The *id* attribute is required.
 - **Col**—Each Row contains a subelement known as a Col, which is expressed as character data. The subelement Col has one attribute, *id*, which is expressed as a character value. This is the same ID value that is used in the MOAttribute. The *id* attribute is required.

SOAP Interface Adapter Implementation

The SOAP interface adapter is a web service application within Tomcat that specifies an external interface. This section provides more detail about the structure of the document interchange between the SOAP adapter and a client-side program.

One global issue for the external interface is that all documents covered here are defined as *well-formed* but not *verified*. This means that the schema is not an embedded part of the XML document. If you embed the schema in a document, parser packages are used to validate the structure of the document. However, this impedes the transition to XML schemas, should schemas be desired by other customers. The client side can still use the schema, included in this document, to perform validation.

Cisco BTS 10200 Softswitch WSDL Code

This section describes the system Web Services Description Language (WSDL) file for the SOAP interface in the BTS 10200. This WSDL code applies to BTS 10200, Release 6.0.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.cisco.com/BTS10200/i01" xmlns:apac
hesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.cisco.com/BTS1020
0/i01" xmlns:intf="http://www.cisco.com/BTS10200/i01" xmlns:wSDL="http://schemas
.xmlsoap.org/wSDL/" xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/" xmlns
:xsd="http://www.w3.org/2001/XMLSchema">
<!--
Copyright (c) 2002-2006 by Cisco Systems, Inc.
-->
<!--WSDL created by Apache Axis version: 1.4
Built on Jul 23, 2006 (06:38:00 CST)-->
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://www.cisco.com/B
TS10200/i01" xmlns="http://www.w3.org/2001/XMLSchema">
<element name="request">
<complexType>
<sequence>
<element name="xmlstring" type="xsd:string"/>
<element name="key" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="requestResponse">
<complexType>
```

```

    <sequence>
      <element name="requestReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<complexType name="BtsSoapException">
  <sequence>
    <element name="error_code" type="xsd:int"/>
    <element name="error_string" nillable="true" type="xsd:string"/>
  </sequence>
</complexType>
<element name="fault" type="impl:BtsSoapException"/>
<element name="login">
  <complexType>
    <sequence>
      <element name="user" type="xsd:string"/>
      <element name="password" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="loginResponse">
  <complexType>
    <sequence>
      <element name="loginReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="logout">
  <complexType>
    <sequence>
      <element name="key" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="logoutResponse">
  <complexType/>
</element>
<element name="getCommandDoc">
  <complexType>
    <sequence>
      <element name="noun" type="xsd:string"/>
      <element name="verb" type="xsd:string"/>
      <element name="key" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="getCommandDocResponse">
  <complexType>
    <sequence>
      <element name="getCommandDocReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="getExtCommandDoc">
  <complexType>
    <sequence>
      <element name="noun" type="xsd:string"/>
      <element name="verb" type="xsd:string"/>
      <element name="key" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
<element name="getExtCommandDocResponse">
  <complexType>

```

```

    <sequence>
      <element name="getExtCommandDocReturn" type="xsd:string" />
    </sequence>
  </complexType>
</element>
</schema>
</wsdl:types>

<wsdl:message name="getExtCommandDocRequest">
  <wsdl:part element="impl:getExtCommandDoc" name="parameters" />
</wsdl:message>

<wsdl:message name="BtsSoapException">
  <wsdl:part element="impl:fault" name="fault" />
</wsdl:message>

<wsdl:message name="requestRequest">
  <wsdl:part element="impl:request" name="parameters" />
</wsdl:message>

<wsdl:message name="loginRequest">
  <wsdl:part element="impl:login" name="parameters" />
</wsdl:message>

<wsdl:message name="getCommandDocRequest">
  <wsdl:part element="impl:getCommandDoc" name="parameters" />
</wsdl:message>

<wsdl:message name="logoutResponse">
  <wsdl:part element="impl:logoutResponse" name="parameters" />
</wsdl:message>

<wsdl:message name="logoutRequest">
  <wsdl:part element="impl:logout" name="parameters" />
</wsdl:message>

<wsdl:message name="getExtCommandDocResponse">
  <wsdl:part element="impl:getExtCommandDocResponse" name="parameters" />
</wsdl:message>

<wsdl:message name="getCommandDocResponse">
  <wsdl:part element="impl:getCommandDocResponse" name="parameters" />
</wsdl:message>

<wsdl:message name="requestResponse">

```



```

        <wsdl:part element="impl:requestResponse" name="parameters" />
    </wsdl:message>

    <wsdl:message name="loginResponse">
        <wsdl:part element="impl:loginResponse" name="parameters" />
    </wsdl:message>

    <wsdl:portType name="Bts10200Operations">
        <wsdl:operation name="request">
            <wsdl:input message="impl:requestRequest" name="requestRequest" />
            <wsdl:output message="impl:requestResponse" name="requestResponse" />
            <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException" />
        </wsdl:operation>

        <wsdl:operation name="login">
            <wsdl:input message="impl:loginRequest" name="loginRequest" />
            <wsdl:output message="impl:loginResponse" name="loginResponse" />
            <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException" />
        </wsdl:operation>

        <wsdl:operation name="logout">
            <wsdl:input message="impl:logoutRequest" name="logoutRequest" />
            <wsdl:output message="impl:logoutResponse" name="logoutResponse" />
            <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException" />
        </wsdl:operation>

        <wsdl:operation name="getCommandDoc">
            <wsdl:input message="impl:getCommandDocRequest" name="getCommandDocRequest" />
            <wsdl:output message="impl:getCommandDocResponse" name="getCommandDocResponse" />
            <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException" />
        </wsdl:operation>

        <wsdl:operation name="getExtCommandDoc">
            <wsdl:input message="impl:getExtCommandDocRequest" name="getExtCommandDocRequest" />
            <wsdl:output message="impl:getExtCommandDocResponse" name="getExtCommandDocResponse" />
            <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException" />
        </wsdl:operation>
    </wsdl:portType>
</wsdl:binding>
</wsdl:service>

```

```

    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="bts10200SoapBinding" type="impl:Bts10200Operations">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="request">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="requestRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="requestResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="BtsSoapException">
      <wsdlsoap:fault name="BtsSoapException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="login">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="loginRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="loginResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="BtsSoapException">
      <wsdlsoap:fault name="BtsSoapException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="logout">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="logoutRequest">
      <wsdlsoap:body use="literal"/>

```

```
</wsdl:input>
<wsdl:output name="logoutResponse">
  <wsdlsoap:body use="literal"/>
</wsdl:output>
<wsdl:fault name="BtsSoapException">
  <wsdlsoap:fault name="BtsSoapException" use="literal"/>
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getCommandDoc">
  <wsdlsoap:operation soapAction="" />
  <wsdl:input name="getCommandDocRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getCommandDocResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="BtsSoapException">
    <wsdlsoap:fault name="BtsSoapException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getExtCommandDoc">
  <wsdlsoap:operation soapAction="" />
  <wsdl:input name="getExtCommandDocRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getExtCommandDocResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="BtsSoapException">
    <wsdlsoap:fault name="BtsSoapException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
```

```
</wsdl:binding>
<wsdl:service name="Bts10200OperationsService">
  <wsdl:port binding="impl:bts10200SoapBinding" name="bts10200">
    <wsdlsoap:address location="https://localhost/axis/services/bts10200"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```