UPS Developer API Technologies for Web Services

Web Services are powerful technologies that let business applications in different enterprises communicate directly with each other. For example, a software application program that processes orders for a mail order retailer can use Web Services to communicate with software applications at UPS that automatically schedule the shipment for new orders.

Web Services are governed by standards bodies which include, but are not limited to, W3C and OASIS. They are not limited to particular vendors and are available to any software application. Applications created for one environment (such as Microsoft Windows) can seamlessly communicate with applications in a different environment (such as Linux) without worrying about incompatibility of the different environments.

Web Services are supported by a wide variety of software development environments, so virtually all software developers can easily add Web Services features to their applications.

In effect, Web Services create a World Wide Web, but for computer applications instead of people. With Web Services, communications between enterprises happens rapidly, efficiently, and reliably.

Two technologies make up the core of Web Services—the Extensible Markup Language (XML) and the Simple Object Access Protocol (SOAP). A third technology, Web Services Definition Language (WSDL) uses XML and SOAP to define specific Web Services.

This section concludes by describing security and error reporting for Web Services.

Extensible Markup Language (XML)

The Extensible Markup Language (XML) is a standard governed by the World Wide Web Consortium, the governing body for web standards and guidelines. XML provides a way to identify the structure of content within a document. Figure 1 shows how a simple XML document could describe a book.

As the figure illustrates, XML distinguishes different parts of a document with labels known as tags. Tags in the example include <book>, <title>, <author>, <firstname>, etc. In this example the publisher for the book is John Wiley and Sons.

```
<?xml version="1.0" encoding="UTF-8" ?>
<book>
    <title>
        HTTP Essentials: Protocols for Secure, Scaleable Web Sites
    </title>
    <author>
        <firstname>
            Stephen
        </firstname>
        <lastname>
            Thomas
        </lastname>
    </author>
    <publisher>
        John Wiley and Sons
    </publisher>
    <year>
        2001
    </year>
    <isbn>
        0-471-398233
```

Figure 1: XML identifies the structure of documents, as in this document describing a book.

A great deal of information on XML is available on the Internet. A good starting point is the World Wide Web Consortium's main page on XML at http://www.w3.org/XML/.

Simple Object Access Protocol (SOAP)

While XML defines the information that Web Services exchange, the Simple Object Access Protocol (SOAP) defines the methods that Web Services use to transfer those documents. The SOAP standard defines several different approaches for sending XML documents, but most Web Services (including those from UPS) rely on a single approach. That method uses the Hypertext Transfer Protocol (HTTP) to send a message from a Web Services client to a server. The server replies in the HTTP response. Figure 2 shows an example of a SOAP header portion of a web services message.

SOAP, like XML, is governed by the World Wide Web Consortium. More information can be found on the Internet at the W3C's XML Protocol Working Group's page, located at http://www.w3.org/standards/xml/

This security header block provides a mechanism for attaching security-related information targeted at a specific recipient in the form of a SOAP actor/role. UPSSecurity is a container element which provides the user access verification for the API Web Service.

```
<envr:Envelope xmlns:auth="http://www.ups.com/schema/xpci/1.0/auth"
    xmlns:upss="http://www.ups.com/XMLSchema/XOLTWS/UPSS/v1.0"
    xmlns:envr="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:common="http://www.ups.com/XMLSchema/XOLTWS/Common/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsf="http://www.ups.com/schema/wsf">
<envr:Header>
<upss:UPSSecurity>
<upss:UsernameToken>
<upss:Username></upss:Username>
<upss:Password></upss:Password>
</upss:UsernameToken>
<upss:ServiceAccessToken>
<upss:AccessLicenseNumber></upss:AccessLicenseNumber>
</upss:ServiceAccessToken>
</upss:UPSSecurity>
</env:Body>
</env:Envelope>
```

Figure 2: A sample of the UPSSecurity header portion of a web services message which structures its content as an XML document.

Web Services Definition Language (WSDL)

XML and SOAP are general technologies used widely for many different purposes. The technology that ties them specifically to Web Services is the Web Services Definition Language (WSDL). Enterprises that make Web Services available to other enterprises describe those services using WSDL. In effect, WSDL acts a service contract: it defines exactly what services the enterprise offers and how clients should access those services.

WSDL documents are XML documents which conform to a specific structure. Figure 3 shows a sample WSDL document. The current version of the specification for WSDL (version 1.1) is available as a draft submitted to the World Wide Web Consortium. It can be found on their web site at http://www.w3.org/TR/wsdl.

Although WSDL documents, like all XML documents, are ultimately textual information, they are not primarily intended for humans to read. Instead, WSDL documents are designed to be read by software applications and application development tools. An application tool such as Microsoft’s Visual Studio can import a WSDL document and automatically generate software classes that access the Web Services the WSDL defines. Developers then add these classes to their applications, giving the programs the ability to use Web Services.

Some WSDL documents are published in special directories such as the Universal Description, Discovery, and Integration (UDDI) registry on the Internet. UPS does not currently publish WSDL documents for

UPS Developer APIs in such directories. Instead, UPS delivers the WSDL documents as part of the software development kit.

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions name="Track" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:error="http://www.ups.com/schema/xpci/1.0/error"
    xmlns:common="http://www.ups.com/XMLSchema/XOLTWS/Common/v1.0"
    xmlns:trk="http://www.ups.com/XMLSchema/XOLTWS/Track/v1.0"
    xmlns:tns="http://www.ups.com/XMLSchema/XOLTWS/Track/v1.0/local"
    targetNamespace="http://www.ups.com/XMLSchema/XOLTWS/Track/v1.0/local">
    <xsd:import namespace="http://www.ups.com/XMLSchema/XOLTWS/Common/v1.0"
        schemaLocation="common.xsd" />
    <xsd:import namespace="http://www.ups.com/XMLSchema/XOLTWS/Error/v1.0"
        schemaLocation="error2.xsd" />
    <xsd:import namespace="http://www.ups.com/XMLSchema/XOLTWS/Track/v1.0"
        schemaLocation="track.xsd" />
    <wsdl:types />
    <wsdl:message name="TrackInput">
        <wsdl:part name="Body" element="trk:TrackRequestDeliveryInterceptRequest" />
    </wsdl:message>
    <wsdl:message name="TrackOutput">
        <wsdl:part name="Body" element="trk:TrackResponseDeliveryInterceptResponse" />
    </wsdl:message>
    <wsdl:message name="TrackError">
        <wsdl:part name="TrackError" element="error:Errors" />
    </wsdl:message>
    <wsdl:portType name="TrackPortType">
        <wsdl:operation name="Track">
            <wsdl:input name="TrackRequestDeliveryInterceptRequest" message="tns:TrackInput" />
            <wsdl:output name="TrackResponseDeliveryInterceptResponse"
                message="tns:TrackOutput" />
            <wsdl:fault name="TrackError" message="tns:TrackError" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="TrackBinding" type="tns:TrackPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="Track">
            .
            <wsdl:fault name="TrackError">
                <soap:fault name="TrackError" use="literal" />
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="Track">
        <wsdl:port name="TrackPortTypePort" binding="tns:TrackBinding">
            <soap:address location="https://www.developerkits.ups.com/webservices/Track" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Figure 3: WSDL documents are specialized XML documents that define Web Services. Example ONLY.

Securing Web Services

Organizations can offer Web Services using the public Internet, so Web Services standards provide many options for securing those services. Security options can ensure that only authorized parties are able to access Web Services, and they can protect confidential information that may be exchanged as part of Web Services communications. Because the breadth and depth of security options for Web Services are extensive, this subsection only describes security features relevant to UPS Developer APIs.

The most basic security services protect confidential information from eavesdropping by other devices on a network. To provide that protection, the UPS Developer APIs rely on the Secure Sockets Layer 3 (SSL3) protocol. When two systems communicate using SSL, the protocol creates a secure channel between them, and it encrypts all information that they exchange using this channel. The SSL protocol that UPS Developer APIs use is the same protocol used to secure millions of on-line purchases on the Web.

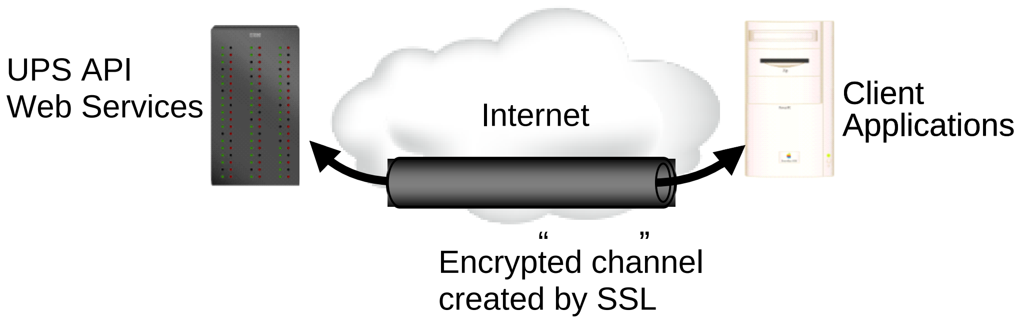

Figure 4: SSL creates a secure channel across a network and protects confidential communications using that channel.

In addition to protecting confidential information, the UPS Developer APIs also ensure that client applications are authorized to access UPS customer information. To gain that authorization, client applications must supply a username, password, and license key in all requests, as the example in Figure 5 shows. UPS corporate applications verify this information before returning sensitive information for the client applications.

```
<?xml version="1.0" encoding="UTF-8" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
    xmlns:upsa="http://www.ups.com/XMLschema/XOLTWS/upssa/v1.0
    xmlns:common="http://www.ups.com/XMLschema/XOLTWS/Common/v1.0
    xmlns:upss="http://www.ups.com/XMLschema/XOLTWS/UPSS/v1.0"
    xmlns:wsf="http://www.ups.com/schema/wsf">
    <env:Header>
        < upss:UPSSecurity>
            <upss:UsernameToken >
                < upss:Username></upss:Username >
                < upss:Password></upss:Password >
            </upss:UsernameToken >
            <upss:ServiceAccessToken >
                <upss:AccessLicenseNumber></upsss:AccessLicenseNumber>
            </upss:ServiceAccessToken >
        </upss:UPSSecurity>
    </env:Header>
    <env:Body>
        <!--  the content of the message goes here  -->
    </env:Body>
</env:Envelope>
```

Figure 5: UPS Web Service requests must include a Username, Password, and AccessLicense.

Included in the WSDL files that document the specific Web Services are the appropriate messages that client applications can use to pass the security credentials as SOAP header elements. Development tools can import the WSDL document and automatically format the request messages appropriately.

Indicating Errors in Client Application Requests

When an error occurs in a client application's request, UPS Developer APIs report that error using the standard SOAP message format. That message format defines a specific message type, known as a *fault*, for error reporting. The essential components of a fault message are the faultcode, faultstring, faultactor, and detail.

The faultcode element can contain one of four values to indicate the type of error that the UPS Developer APIs encountered.

- VersionMismatch: The SOAP message that the client application sent used a version of the SOAP protocol that the UPS Developer APIs could not understand.
- MustUnderstand: The SOAP message that the client application sent included an element in the header that the UPS Developer APIs could not understand.

- Client: The request that the client application sent was not valid.
- Server: Although the client application's request did not have any errors itself, the UPS Developer APIs encountered an error when trying to process it.

The faultstring element contains a textual description of the error.

The faultactor element can indicate which system detected or generated the error. If present, it contains a Uniform Resource Identifier (URI) for that system.

The detail element contains more information about the error. It includes a specific error code and a textual description for that code.

> Note: UPS encourages application developers to display the description of any unexpected errors or warnings to the user. This information can be invaluable when diagnosing problems, and will normally be required by UPS Technical Support.
>
> It will be extremely helpful if the developer implements and maintains logs of all transactions and activity, including errors or warnings.