Sabre Scribe Developer – Access to Sabre Profiles

STUDENT GUIDE

Table of Contents

Edition 1.0 (August 2013).

O b j e c t i v e s

This workshop is designed to explain how to use Sabre Scribe scripts to interact with the Sabre Profiles system. At the end of this workshop, you will be able to:

- Utilize the new EPS: construct to create Sabre Profiles web services request messages from a Sabre script.
- Use the 6 Sabre Profiles web service request types that Sabre Scribe is capable of communicating with: ProfileSearchRQ, ProfileReadRQ, ProfileCreateRQ, ProfileUpdateRQ, ProfileDeleteRQ and ProfileToPNRRQ.
- Read data contained in Sabre Profiles response messages that are received in response to request messages.
- Locate reference information on the structure of Sabre Profiles web services messages.
- Use the emulator.log file to troubleshoot problems with writing Sabre Profiles web service requests and reading Sabre Profiles web service responses.

Please list any additional objectives you have below and bring them to the attention of the instructor for discussion:

Overview

Sabre Profiles is Sabre's profile information system. The Sabre Profiles system stores data in a relational database. Access to this database utilizes web service requests from a client application in order to receive responses from the Sabre Profiles database. Sabre Travel Network agency customers may access Sabre Profiles either via Sabre Red Workspace point of sale or directly by consumption of the Sabre Profiles web services via subscription to that service.

This workshop focuses on access to Sabre Profiles via Sabre Red Workspace point of sale since it addresses utilization of Sabre Scribe scripts which are run from Sabre Red Workspace.

In order to access Sabre Profiles via Sabre Red Workspace, an agency must have completed migration to Sabre Profiles from the Sabre STARs system. If your agency has not already migrated to Sabre Profiles then contact your Sabre account team to discuss Sabre Profiles.

This workshop covers creating scripts which can access the Sabre Profiles system but it does not provide training on the Sabre Profiles system itself. To get the most from this workshop, it is necessary to have knowledge of the Sabre Profiles system including knowledge of such concepts as templates, profile subject areas and data elements, custom formats, PNR Builders (filters) and profile associations and Copy To PNR functionality.

This workshop will address how to create Sabre Profiles web service XML requests using the special syntax developed for Sabre Scribe. A brief introduction of how web services function and XML schemas is provided here; however, Sabre Scribe developers, who wish to write scripts that will utilize the functionality to communicate to the Sabre Profiles database, will benefit from a full understanding of XML and web services functionality and familiarity with reading XML schemas which cannot be conveyed here.

Sabre Profiles reference tools such as Sabre Profiles schemas and sample XML messages are available from Sabre's Developer Resource Center. You must have an account with an assigned user name and password to access the center.
https://drc.sabre.com/

It is recommended that Sabre Scribe script writers who wish to write scripts to access Sabre Profiles have access to an XML reading software to simplify the reading of the

Sabre Profiles schemas. Such software may be acquired through third party sources (e.g. Altova XML Spy); Sabre does not supply XML reading software.

Description

Sabre Scribe allows scripts to interact with data in the Sabre Profiles database. Sabre Scribe scripts utilize a specialized variation of Sabre Profiles external web services messages. The Sabre Red Workspace runtime software creates much of the web services message (such as the SOAP envelope and headers and session create) behind the scenes; the script developer only needs to create the service request portion of the message and read from the response.

When the scripts are executed, the Sabre Scribe runtime converts the scribe syntax into actual web services messages. Sabre Scribe scripts can send request messages and receive and process response messages from the Sabre Profiles database in order to read existing data in profiles, create profiles and update profiles or move profile data into the Sabre emulator to create PNRs.

Linear Sabre Profiles Formats

Linear formats supported for Sabre Profiles may be entered by a script using the normal method of sending a format to the emulator without utilizing the new syntax used to create a web services message.

A new setting for [@SWITCHES] may be set to "POINTCLICK" which allows a feature of Sabre Red Workspace to remain active during the execution of scripts which enables the point of sale to route the linear Sabre Profiles formats to the necessary processor to obtain responses from the Sabre Profiles database.

For example, if a script only needs to perform a blind profile move using the linear format N*(profile name)§NM then this can be accomplished by setting the [@SWITCHES] system variable to "POINTCLICK" and then entering the linear format from the script using the standard syntax as in the example below:

`»N*(`*`profile name`*`)§NM{ENTER}«`

Example:

```
»N*ABCCORP§NM{ENTER}«
```

The default value of this aspect of [@SWITCHES] is "NOPOINTCLICK". This setting may either be set manually within the script or will be the setting by default if [@SWITCHES] is not set to "POINTCLICK". This setting affects whether Sabre

Red Workspace Point and Click functionality remains active during the execution of scripts.

In addition to affecting the way Sabre Profiles linear formats are processed when issued by a script, setting the value of [@SWITCHES] equal to “POINTCLICK” also means that when formats are issued by a script which normally receive responses containing clickable areas (i.e. the Point and Click feature of Sabre Red Workspace) those clickable areas will be present and remain functional in responses received to formats issued by the script.

A Brief Introduction to Web Services

Web services let computers talk to one another over the Internet, allowing computer programs to exchange information by eliminating barriers such as different hardware platforms, software languages, and operating systems that usually make different programs incompatible.

Web services are automated information services that are conducted over the Internet, using standardized technologies and formats/protocols that simplify the exchange and integration of large amounts of data over the Internet.

For our purposes, the essential concept to understand regarding web services is that they allow you to access the system that you want to communicate with (AKA the service provider, which is, in our case, Sabre’s Sabre Profiles relational database) by sending messages through the Internet that must follow a precise format. The message that you send is a request. In reply to the message that you send, the service provider sends another message which is the response message.

The language in which the message is composed is Extensible Markup Language (XML) which follows a Simple Object Access Protocol (SOAP.) A guide, known as a schema, is published by the service provider to help you know the requirements of the message. The schema is written using Web Services Description Language (WSDL).

In order to write scripts to access Sabre Profiles, it is not essential to have a deep knowledge of XML or SOAP or WSDL beyond an understanding of how XML messages are structured and how to interpret the schemas.

Brief definitions of some of the terms used above are provided here.

WSDL

Web Services Description Language (WSDL) is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically.

XML

Extensible Markup Language (XML) is a flexible language for creating common information formats and sharing both the format and content of data over the Internet and elsewhere. XML is a formatting language recommended by the World Wide Web Consortium (W3C).

SOAP

A Simple Object Access Protocol (SOAP) is a way for a program running in one kind of operating system (such as Windows 7) to communicate with a program in the same or another kind of an operating system (such as Linux) by using the World Wide Web's Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML) as the mechanisms for information exchange.

Schema

An XML schema defines the structure of an XML document. An XML schema defines things such as which data elements and attributes can appear in a document; how the data elements relate to one another; whether an element is empty or can include text; which types of data are allowed for specific data elements and attributes; and what the default and fixed values are for elements and attributes. A schema is also a description of the data represented within a database. The format of the description varies but includes a table layout for a relational database or an entity-relationship diagram.

The service provider makes several types of services available that allow you to accomplish the different tasks involved with utilizing the service. Each service has a specific name and the message has a specific format and requirements that must be adhered to. You, as the service user, select the type of service you want to use based on the task you wish to perform and then send a request message which follows the schema for that service type. The service provider's response message also has a specific format that is adhered to which simplifies the task of reading the response you receive.

The web services messages contain several parts such as the envelope information which provide the destination for the message on the Internet and the identity of the sender of the message and time stamps and conversation IDs that enable the computers involved in the conversation to track the communication as well as the actual service request that will be transacted once the message gets to the right location at the service provider.

In the case of Sabre Scribe scripts which communicate with the Sabre Profiles database, the components of the web service request is automatically created by the Sabre Red Workspace Scribe runtime so, as the developer of the script, the only component you need to be concerned with is the actual request portion of the message and the response.

In Sabre Scribe scripts, the XML message is created using a special Sabre Scribe syntax which condenses the normal XML message into a unique shorthand. The Sabre Red Workspace Scribe runtime application then interprets the syntax of the script and creates the actual web services message which is handed off to the Sabre Red Workspace program which sends it via the Internet to the Sabre Profiles database.

The response is received by Sabre Red Workspace where the contents can be read by the Sabre Scribe script.

The web service message sent to the service provider, as well as the response message, can be seen in a log file which is created whenever a Sabre Red Workspace session is in use. The file is called emulator.log. In order for the log file to capture the web services requests and responses, the emulator logging level in Sabre Red Workspace must be set to LOW. See the section on Diagnostic Tools in this guide for more details.

In order to successfully execute scripts which communicate with the Sabre Profiles database, the end-user must sign into the Sabre point of sale using an EPR from a pseudo city code (PCC) which has been migrated to Sabre Profiles.

The EPR of the end-user at a pseudo city code migrated to Sabre Profiles will contain the following keywords (AKA ICE attributes):

- RolesReadOnlyExternalUser
- EPSExternalUser
- EPSUser
- PnrMoveUser
- TravelPolicyExternalUser
- MySabreWCScribeUser

If any of the above keywords are missing from the EPR, running the scripts will cause request messages to be created but the requests will fail to receive responses in return. The end-user's EPR attributes can be verified by reviewing the emulator.log file which is created each time a Sabre Red Workspace session is opened. The emulator.log file is located in the following directory:

C:\Sabre\Apps\Emulator\Users\{*end-user's Sabre Agent ID*}\emulator.log.

The log contains a section entitled SessionConfig. A list of the keywords appears within the SessionConfig section. An example of the SessionConfig section of the emulator.log file appears in Appendix A.

If any of the required keywords are absent and the EPR/TJR is from a pseudo city code that has been migrated to Sabre Profiles please contact your Sabre account team. Sabre account team members will work with the Sabre Profiles migration team to have the necessary attributes added to the EPR.

Sabre Profiles web services available via Sabre Scribe scripts

Communication to the Sabre Profiles database is conducted by means of web services messages which are embedded in scripts. There are six services that may be utilized by Sabre Scribe scripts:

- **Sabre_OTA_ProfileReadRQ** – used to read the contents of a Sabre profile, template, filter or format
- **Sabre_OTA_ProfileSearchRQ** – used to locate a Sabre profile, template, filter or format using various data elements
- **Sabre_OTA_ProfileCreateRQ** – used to create a Sabre Profile, template, filter or format
- **Sabre_OTA_ProfileDeleteRQ -** used to delete or restore a Sabre Profile, template, filter or format
- **Sabre_OTA_ProfileUpdateRQ -** used to add, delete or change information in an existing Sabre Profile, template, filter or format.
- **Sabre_OTA_ProfileToPNRRQ** – used to move Sabre Profile data into the Emulator work area (copy to PNR)

Once a request is sent from the script to the Sabre Profiles database, a response message is received.

If the request message was formatted correctly according to the XML schemas, a response message containing data is sent back from the Sabre Profiles database and received by the Sabre point of sale. If the request message did not comply with the XML schemas an error message is received.

The response types are as follows:

- **Sabre_OTA_ProfileReadRS**
- **Sabre_OTA_ProfileSearchRS**
- **Sabre_OTA_ProfileCreateRS**
- **Sabre_OTA_ProfileDeleteRS**
- **Sabre_OTA_ProfileUpdateRS**
- **Sabre_OTA_ProfileToPNRRS**

The "Sabre_OTA_" prefix is optional when indicating the service request type being written to or response type being read from when using the Sabre Profiles client feature in Scribe. In most examples in this guide, the "Sabre_OTA" prefix has been omitted.

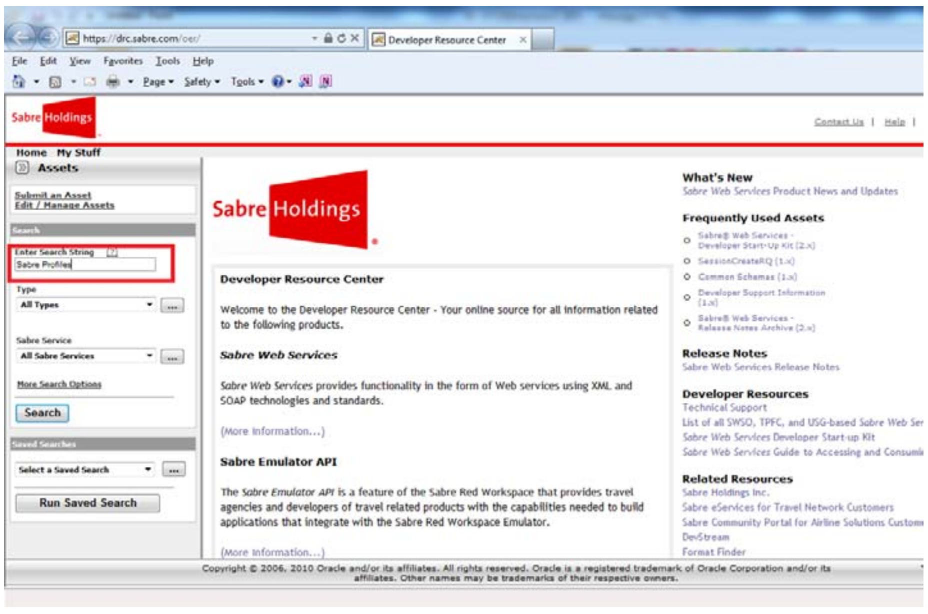
Each web service request may consist of several mandatory and optional components. Reviewing the XML schemas will provide the information detailing which elements are required and which optional elements the message may include as well as the order, parameters and iterations of these elements.

The schemas for all the services are available in the Developer Resource Center (see section on Reference Information).

Reference Information

Sabre's Developer Resource Center provides access to information about web services including the Sabre Profiles web services. Developers may go to the Developer Resource Center in order to view and download information on Sabre Profiles web services such as a guide on the use of the services entitled *Sabre Profiles Technical User Guide.pdf*, the actual schemas for each service request and response and sample XML code. It is necessary to have a username and password provided by Sabre to access the Developer Resource Center.

After completing the log-in page of the Developer Resource Center, a quick way to locate the reference information available on the Sabre Profiles web services is to use the Search widget which appears on the left-side of the screen. Enter 'Sabre Profiles' on the Enter Search String box.


https://drc.sabre.com/oer/

Developer Resource Center

Home My Stuff

Assets

Submit an Asset
Edit / Manage Assets

Search

Enter Search String: Sabre Profiles

Type: All Types

Sabre Service: All Sabre Services

More Search Options

Search

Saved Searches

Select a Saved Search

Run Saved Search

Contact Us | Help |

Developer Resource Center

Welcome to the Developer Resource Center - Your online source for all information related to the following products.

Sabre Web Services

Sabre Web Services provides functionality in the form of Web services using XML and SOAP technologies and standards.

(More information...)

Sabre Emulator API

The *Sabre Emulator API* is a feature of the Sabre Red Workspace that provides travel agencies and developers of travel related products with the capabilities needed to build applications that integrate with the Sabre Red Workspace Emulator.

(More information...)

What's New
Sabre Web Services Product News and Updates

Frequently Used Assets
- Sabre® Web Services - Developer Start-Up Kit (2.x)
- SessionCreateRQ (1.x)
- Common Schemas (1.x)
- Developer Support Information (1.x)
- Sabre® Web Services - Release Notes Archive (2.x)

Release Notes
Sabre Web Services Release Notes

Developer Resources
Technical Support
List of all SWSO, TPFC, and USG-based Sabre Web Ser
Sabre Web Services Developer Start-up Kit
Sabre Web Services Guide to Accessing and Consumi

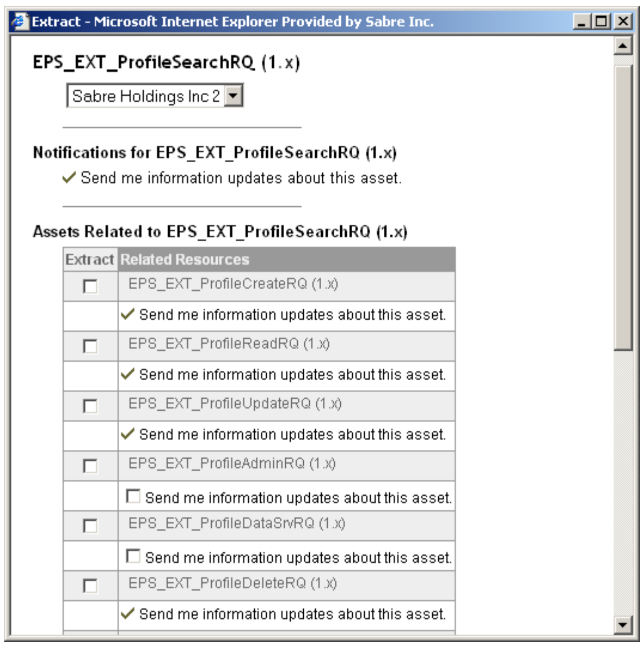
Related Resources
Sabre Holdings Inc.
Sabre eServices for Travel Network Customers
Sabre Community Portal for Airline Solutions Custom
DevStream
Format Finder

Copyright © 2006, 2010 Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The files that will be of interest to Sabre Scribe script developers are those entitled EPS_EXT_(service request type) as listed in the preceding section of this guide but without the prefix Sabre_OTA_.

Clicking on the name of a service you are interested in and then clicking the Use-Download button allows you to download or access the files for that service as well as related services. The item listed as Sabre Profiles Profile Services provides access to the Technical Documentation.

For example, click on the EPS_EXT_ProfileSearchRQ item from the list and then click on Use – Download. A pop up appears like the one below:


Extract - Microsoft Internet Explorer Provided by Sabre Inc.

EPS_EXT_ProfileSearchRQ (1.x)

Sabre Holdings Inc 2

Notifications for EPS_EXT_ProfileSearchRQ (1.x)

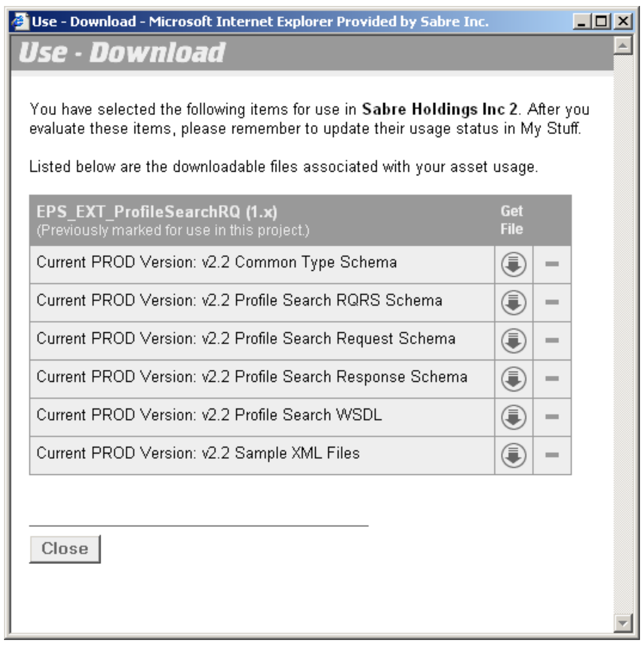
✓ Send me information updates about this asset.

Assets Related to EPS_EXT_ProfileSearchRQ (1.x)

Extract	Related Resources
☐	EPS_EXT_ProfileCreateRQ (1.x)
	✓ Send me information updates about this asset.
☐	EPS_EXT_ProfileReadRQ (1.x)
	✓ Send me information updates about this asset.
☐	EPS_EXT_ProfileUpdateRQ (1.x)
	✓ Send me information updates about this asset.
☐	EPS_EXT_ProfileAdminRQ (1.x)
	☐ Send me information updates about this asset.
☐	EPS_EXT_ProfileDataSrvRQ (1.x)
	☐ Send me information updates about this asset.
☐	EPS_EXT_ProfileDeleteRQ (1.x)
	✓ Send me information updates about this asset.

Click the checkboxes for related services if you also wish to download those files at this time or just click the Next button which appears at the bottom of the window. It will be necessary to scroll down to locate the button.

A pop-up like the one below appears:


Use - Download - Microsoft Internet Explorer Provided by Sabre Inc.

Use - Download

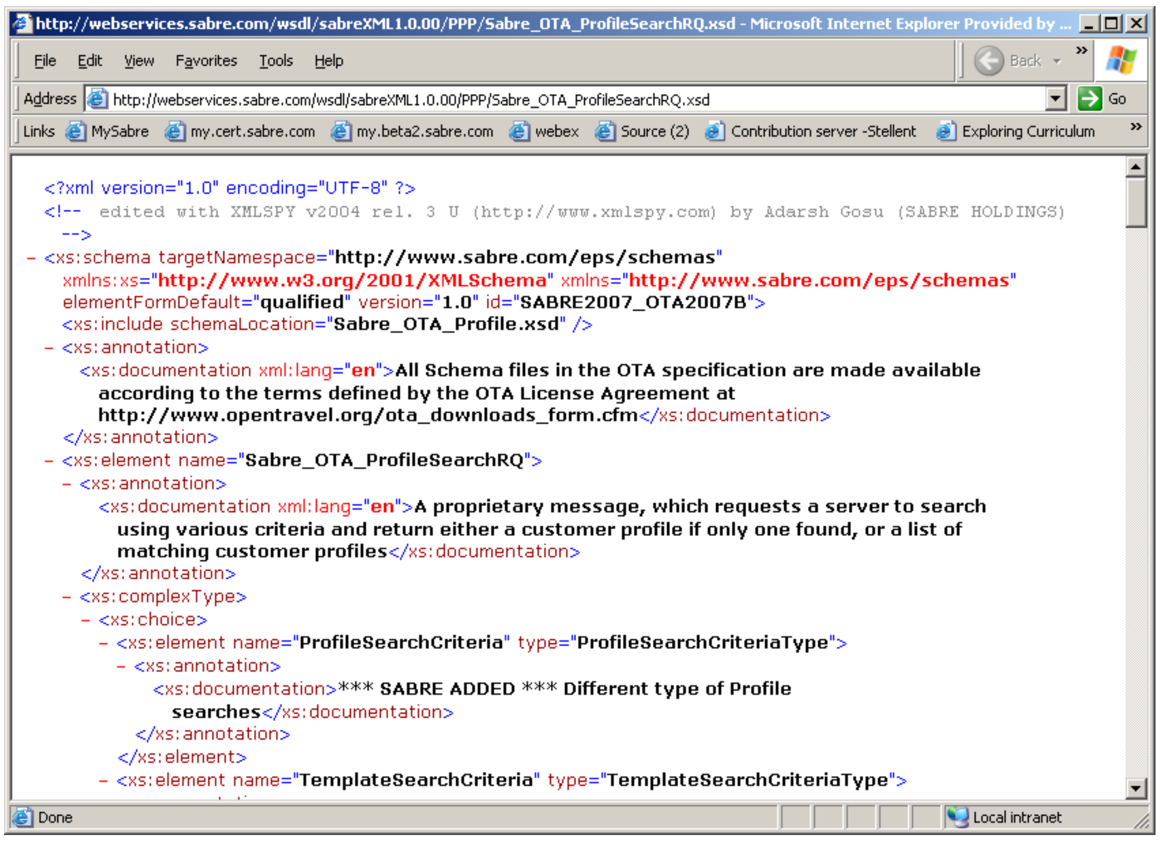
You have selected the following items for use in **Sabre Holdings Inc 2**. After you evaluate these items, please remember to update their usage status in My Stuff.

Listed below are the downloadable files associated with your asset usage.

EPS_EXT_ProfileSearchRQ (1.x) (Previously marked for use in this project.)	Get File	
Current PROD Version: v2.2 Common Type Schema	⬇	–
Current PROD Version: v2.2 Profile Search RQRS Schema	⬇	–
Current PROD Version: v2.2 Profile Search Request Schema	⬇	–
Current PROD Version: v2.2 Profile Search Response Schema	⬇	–
Current PROD Version: v2.2 Profile Search WSDL	⬇	–
Current PROD Version: v2.2 Sample XML Files	⬇	–

Click the arrow in the Get File column to download that resource. It is likely that for developers of Sabre Scribe scripts, the files of interest are the (Service Type) Schema (i.e. the third and fourth files in the screen shot above) and the Sample XML files (i.e. the last file in the screen shot above).

After clicking on the down arrow adjacent to the item you wish to download in the Get File Column the XML appears in a browser window.


http://webservices.sabre.com/wsdl/sabreXML1.0.00/PPP/Sabre_OTA_ProfileSearchRQ.xsd - Microsoft Internet Explorer Provided by ...
File Edit View Favorites Tools Help
Back
Address http://webservices.sabre.com/wsdl/sabreXML1.0.00/PPP/Sabre_OTA_ProfileSearchRQ.xsd
Go
Links MySabre my.cert.sabre.com my.beta2.sabre.com webex Source (2) Contribution server -Stellent Exploring Curriculum
Done
Local intranet

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <!--  edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by Adarsh Gosu (SABRE HOLDINGS)
    -->
- <xs:schema targetNamespace="http://www.sabre.com/eps/schemas"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.sabre.com/eps/schemas"
    elementFormDefault="qualified" version="1.0" id="SABRE2007_OTA2007B">
    <xs:include schemaLocation="Sabre_OTA_Profile.xsd" />
  - <xs:annotation>
      <xs:documentation xml:lang="en">All Schema files in the OTA specification are made available
        according to the terms defined by the OTA License Agreement at
        http://www.opentravel.org/ota_downloads_form.cfm</xs:documentation>
    </xs:annotation>
  - <xs:element name="Sabre_OTA_ProfileSearchRQ">
    - <xs:annotation>
        <xs:documentation xml:lang="en">A proprietary message, which requests a server to search
          using various criteria and return either a customer profile if only one found, or a list of
          matching customer profiles</xs:documentation>
      </xs:annotation>
    - <xs:complexType>
      - <xs:choice>
        - <xs:element name="ProfileSearchCriteria" type="ProfileSearchCriteriaType">
          - <xs:annotation>
              <xs:documentation>*** SABRE ADDED *** Different type of Profile
                searches</xs:documentation>
            </xs:annotation>
          </xs:element>
        - <xs:element name="TemplateSearchCriteria" type="TemplateSearchCriteriaType">
```

You may then save this file to your computer and then use it as is for reference or open it with an XML reading software if you have one.

The Request schemas will tell you how to structure a message which utilizes that service type. The Response schemas will tell you how the reply message will be structured.

See Appendix B for an example of how the ProfileSearchRQ schema looks in the Schema Design View of third-party XML reading software.

The software display makes clear the node and child node relationships in the XML request message.

A sample of the actual XML message to send a ProfileSearchRQ might look like this (envelope and message header information has been omitted):

```
<Sabre_OTA_ProfileSearchRQ Version="1.0"
xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemasWSDL\Sabre_OTA_ProfileSearchRQ.xsd"
xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

       <ProfileSearchCriteria ProfileNameOnly="N" >

              <TPA_Identity ClientContextCode="MYS" ClientCode="TN" DomainID="*" >

                   <ProfileName="srphillips" ProfileTypeCode="ALL" >

              </TPA_Identity>

       </ProfileSearchCriteria>

</Sabre_OTA_ProfileSearchRQ>
```

The values that are permitted in fields where the child name ends in “-Code” (e.g. ClientContextCode or ProfileTypeCode) are found in the Control data of the Sabre Profiles reference material. For example, the ProfileTypeCode.xml shows the 7 values that can be used in the ProfileTypeCode attribute of the TPA_Identity node: AGT, AGY, ALL, CRP, OPX, GRP and TVL.

In order to send this ProfileSearchRQ message by script the syntax of the script would be as follows:

```
[REQ] = "EPS:req.ProfileSearchRQ.ProfileSearchCriteria."
WRITE F="EPS:req.ProfileSearchRQ.@Version" "2.2"
WRITE F=[REQ] + "@ProfileNameOnly" "N"
WRITE F=[REQ] + "TPA_Identity.@ProfileName" "srphillips"
WRITE F=[REQ] + "TPA_Identity.@DomainID" "*"
WRITE F=[REQ] + "TPA_Identity.@ProfileTypeCode" "ALL"
WRITE F=[REQ] + "TPA_Identity.@ClientCode" "TN"
WRITE F=[REQ] + "TPA_Identity.@ClientContextCode" "MYS"
```

The syntax of the above code is explained in the section on The Sabre Profile Device.

Exercise 1

Suppose you wanted to write a script to determine what data elements of a profile were pre-selected for Copy To PNR by a filter associated to that profile. What Sabre Profile schema would you refer to as a guide to assist with development of the script?

____

Answers to the exercises are found in Appendix F of this guide.

Sabre Scribe scripts use a special syntax to create the web service XML messages which communicate to the Sabre Profiles database. The tags of the normal XML message become nodes in the script syntax.

An "EPS:" WRITE/READ device exists for creating and reading Sabre Profiles web services requests and responses. The device is used with WRITE to create a request message and with READ to read a response message.

Syntax:

WRITE F= "EPS:req.{*name of Sabre Profiles web service request being used*}**.**{*node of message to populate*}**.**{*sub node of preceding node to populate*}**" "***value to populate in node***"**

WRITE F= "EPS:req.{*name of Sabre Profiles web service request being used*}**.**{*node of message to populate*}**.@**{*attribute of preceding node to populate*}**" "***value to populate in attribute***"**

READ F= "EPS:rsp.{*name of Sabre Profiles web service response being processed*}**.**{*node of message being read*}**.**{*sub-node of preceding node being read*}**" [***variable to contain the value in the node***]**

Examples:

```
WRITE F="EPS:req.Sabre_OTA_ProfileToPNRRQ.ProfilePath.Traveler.Customer.PersonName.
     SurName" "SMITH"
```

Note: As stated earlier, the "Sabre_OTA" portion of the name of the service may be omitted so the above syntax may also be written as below:

```
WRITE F="EPS:req.ProfileToPNRRQ.ProfilePath.Traveler.Customer.PersonName.SurName" "SMITH"

WRITE F="EPS:req.ProfileToPNRRQ.ProfilePath.Traveler.Customer.PersonName.SurName" [SUR_NAME]

WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@ProfileTypeCode" "TVL"

READ F="EPS:rsp.ProfileReadRS.Profile.TPA_Identity.@ProfileName" [PROFILENAME]
```

In order to compose a request message in a script, the XML schemas are consulted and then reproduced using the above syntax in place of XML tags.

The case (i.e. upper or lowercase) conventions from the XML schema node names must be replicated exactly when writing the Sabre Scribe code.

A convenience technique, used in many of the example scripts provided in this section, is to assign a portion of the request or response message to a variable which can then be used in subsequent WRITE or READ commands to avoid repeating that portion of the message.

In the following example, the web service being utilized by the script is the Sabre_OTA_ProfileSearchRQ request and the response that will be read is the Sabre_OTA_ProfileSearchRS response. The request type and response type are assigned to a variable which can then be used to concatenate the entire value with fewer keystrokes:

```
[REQ] = "EPS:req.ProfileSearchRQ.ProfileSearchCriteria."

[RSP] = "EPS:rsp.ProfileSearchRS.ProfileInfo."
```

A subsequent WRITE command may then be written as:

```
WRITE F=[REQ] + "TPA_Identity.@ProfileName" [PROFILENAME]
```

Or when reading from the response:

```
READ F=[RSP] + "Profile.Traveler.Customer.PersonName.GivenName" [FIRST_NAME]
```

When utilizing the technique described above, do not separate the period (.) and at symbol (@) used prior to an attribute name as it causes an error in the request message generated to the Sabre Profiles database.

In other words, you may do this:

```
[REQ]="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.@"
WRITE F=[REQ]+"ProfileNameOnly" "No"
```

Or:

```
[REQ]="EPS:req.ProfileSearchRQ.ProfileSearchCriteria"
WRITE F=[REQ]+".@ProfileNameOnly" "No"
```

But not this:

```
[REQ]="EPS:req.ProfileSearchRQ.ProfileSearchCriteria."
WRITE F=[REQ]+"@ProfileNameOnly" "No"
```

Creating the Sabre Profiles Request Message

By using the syntax introduced in the preceding section, the request portion of the XML message is created using several WRITE commands. The remainder of the XML message is created automatically by Sabre Scribe Runtime (i.e. the namespace attributes {schemaLocation, xmlns} are pre-populated automatically by Sabre Scribe.)

The entire request portion is created with a series of WRITE commands before the message is sent to the Sabre Profiles database. The message must contain all required elements as indicated by the XML schemas as well as any optional elements that are desired.

Let's consider the following Sabre Profiles request object (a SOAP message):

```
 <Sabre_OTA_ProfileSearchRQ Version="1.0"
   xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemasWSDL\Sabre_OTA_ProfileSearchRQ.xsd"
   xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ProfileSearchCriteria ProfileNameOnly="No">
        <TPA_Identity ClientContextCode=”TN” ClientCode="MYS" DomainID="L3OB"  ProfileName="MYSSTAR"
         ProfileTypeCode="TVL">
        </TPA_Identity>
    </ProfileSearchCriteria>
</Sabre_OTA_ProfileSearchRQ>
```

This message needs to be created by accessing the "request" property of the "EPS:" device. Note that the namespace attributes (schemaLocation, xmlns) are pre-populated automatically by Scribe. The following code sets the remaining values:

```
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.@ProfileNameOnly" "No"
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@ProfileName" "MYSSTAR"
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@DomainID" "L3OB"
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@ProfileTypeCode" "TVL"
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@ClientCode" "MYS"
WRITE F="EPS:req.ProfileSearchRQ.ProfileSearchCriteria.TPA_Identity.@ClientContextCode" "TN"
```

(Note that the optional "Sabre_OTA_" prefix of the root node is not included in the above example.)

Although the node names (including case) must be replicated exactly from the schemas in order to create a successful request, the order that elements of the message are added by the script does not need to match the schema order exactly. For example, the schema for the ProfileSearchRQ shows the attributes of TPA_Identity in the following order: ProfileTypeCode, ClientCode, ClientContextCode, ProfileName, DomainID however, the syntax in the example above may be used to successfully compose the ProfileSearchRQ request message even though the attributes are provided in a different order.

A WRITE command is used for each node, sub node or attribute that you wish to populate in the request message. All mandatory elements of the web services request must be populated as well as any optional elements that you wish to include. It may require several consecutive WRITE statements to complete the message.

When Sabre Scribe is used to create XML messages, the nested tags of the XML structure are reproduced as nodes and child nodes in the Scribe syntax.

For example, suppose the structure of the XML document is as follows:

```
<root>

      <node1>value</node1>

</root>
```

The corresponding Sabre Scribe syntax would be:

```
WRITE F="EPS:req.root.node1" "value"
```

In the following example, the XML tags house attributes within those tags:

```
<root>

      <node 1 attribute1=”value”
       attribute2=”value2></node1>

</root>
```

The corresponding Sabre Scribe syntax would be:

```
WRITE F="EPS:req.root.node1.@attribute1" "value1"
WRITE F="EPS:req.root.node1.@attribute2" "value2"
```

An XML document might contain multiple sibling nodes with the same name. For example, the phone subject area of a profile may contain two (or more) phone numbers.

The portion of the XML message to copy phone numbers to a PNR using the ProfileToPNRRQ may look like this:

```
<Telephone InformationText=”business phone” OrderSequenceNo=”2” LocationTypeCode=”BUS”>
       <FullPhoneNumber>682-605-1234</FullPhoneNumber>
</Telephone>


<Telephone InformationText=”home phone” OrderSequenceNo=”1” LocationTypeCode=”HOM”>
       <FullPhoneNumber>414-123-1234</FullPhoneNumber>
</Telephone>
```

The Sabre Scribe syntax to produce the above XML message assigns an iteration number to each set of telephone related data starting with zero. An example of the code to produce the XML message shown above is as follows (values for these nodes or attributes may be either hard coded or provided from variables or a combination as in the example below):

```
WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[0].@LocationTypeCode" [TEL2TYPE]

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[0].@OrderSequenceNo" "2"

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[0].@InformationText" [TEL2TEXT]

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[0].FullPhoneNumber" [TEL2FULLPHONE]

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[1].@LocationTypeCode" [TEL1TYPE]

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[1].@OrderSequenceNo" "1"

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[1].@InformationText" [TEL1TEXT]

WRITE F=[REQ] + "ProfilePath.Traveler.Customer.Telephone[1].FullPhoneNumber" [TEL1FULLPHONE]
```

Another example of a complete Sabre Profiles web service message (Sabre_OTA_ProfileCreateRQ for building a profile) in XML format appear in Appendix C part 1. The syntax used in a script to compose the same web service request appears in Appendix C part 2.

ClientCode and ClientContextCode values in Sabre Profiles web service messages

The Sabre Profiles database is designed to be used by various types of Sabre customers and from various environments; therefore, settings exist within the Sabre Profiles services to identify the type of customer and environment in which the Sabre Profiles web service will be consumed.

Scripts developed by Sabre Travel Network’s travel agency customers will always be designed to be sent on behalf of Travel Network customers and always from the Sabre point of sale environment (i.e. Sabre Red Workspace). Due to this fact, two settings used as part of the identifying information in all Sabre Profiles web services request messages will always be the same for travel agency customers. These settings are the TPA_Identity attributes of ClientCode and ClientContextCode.

The ClientCode attribute value will always be set to “TN” since all travel agency customers are accessing the Travel Network area of Sabre Profiles. The ClientContextCode value will always be set to “MYS” since all scripts which consume the Sabre Profiles web services will be executed from the Sabre Red Workspace.

Opening a Connection

Once the request message is composed, it needs to be sent to the Sabre Profiles database for processing. The following syntax needs to be used in order to open a web service connection (i.e. send the message)

```
OPEN F="EPS:"
```

Upon sending a request the Sabre Profiles device waits until a response arrives and pre-populates its "response" property. The script may then read the values in the response message.

Exercise 2

Write the Sabre Scribe code to create the request message XML to read a custom format. Include all the required elements in your request message (do not write the code to read the response).

The script should contain a window to allow the end user to enter the required data to identify the format (use the schemas to determine these).

Have the script check to determine if an error was received after sending the request to the Sabre Profiles database and, if so, display a window to advise the end-user that an error was received.

Answers to the exercises are found in Appendix F of this guide.

Processing the Sabre Profiles Response Message

After the OPEN command is used to send the request message to the Sabre Profiles database, a response message is received. The response, like the request, is a SOAP message that can be read using the “EPS:” device.

It is necessary to read the response within the same script as the request message was sent. If you attempt to create the request message in one script and then read the response in a second script (one that is either CHAINed or SPAWNed from the initial script or manually invoked) the READ statements will fail to detect any values from the response.

There are five types of response objects (listed in the Overview section above), each corresponding with a request type (i.e. if your script used the Sabre_OTA_ProfileReadRQ to send a request to read the contents of a profile, the response you will receive will be the Sabre_OTA_ProfileReadRS which will contain the data elements in the profile.).

The response message follows the format describe by the XML schema for that response type.

The response message will contain either a “success” node if the request message was successfully processed by the Sabre Profiles database or contain an "error" node if the request message was not successfully processed.

Errors are normally the result of not correctly following the XML schema for the service used in the request message. The XML schema can be violated by (1) the construction of the WRITE commands in your script used to create the request but also may be caused by (2) the input of variable data by the end-user if the data does not follow the requirements for that data type in the schema.

A request which includes the following WRITE statement would generate an error as a result of the first (1) situation describe above because the @ is missing prior to the attribute Version:

```
WRITE F="EPS:req.ProfileCreateRQ.Version" "2.2"
```

A request which includes the following WRITE statement could also generate an error even though the syntax of the command is correct if the end-user inputs a value of ZZZ for [PCC] because the ProfileCreateRQ schema requires that the data input for the DomainID attribute consists of a 4 character alpha/numeric code.

```
WRITE
F="EPS:req.ProfileCreateRQ.Profile.TPA_Identity.@DomainID"
[PCC]
```

A shorthand syntax exists which enables checking for the presence of an error node in a response message:

```
READ F="EPS:rsp.error" [ERR]

;(note that capitalizing Error in the line above will also work)

IF [ERR] = "TRUE" THEN

    »EPS failed!«


ENDIF
```

The value of [ERR] will be "TRUE" if the "error" node was found in the response message. The value of [ERR] will be "FALSE" if the "error" node was not found in the response message (i.e. the "success" property was found).

*Note that this syntax for locating the presence of an error is a special one as the actual name of the node in a response message when that response contains an error is called Error**s** (not error as in the READ statement above, and furthermore, the actual Errors node is contained within the ResponseMessage node of a response message.*

This special shorthand syntax DOES NOT extend to searching for the presence of the Success node (i.e. you cannot utilize the syntax READ F= "EPS:rsp.success" [ABC] in search of a value of either TRUE or FALSE for [ABC]. Instead, use the method described below for searching for the presence of a node.

If the response message was found to not contain the "error" property then the contents of the response can be read with the "EPS:" device.

In processing the response message you may wish read the contents of the data contained within the nodes or to determine if the response message contains a particular node.

An example of the syntax to read the contents of data within a node is as follows (note that the value of RSP has been previously set to "EPS:rsp.ProfileSearchRQ."

```
READ F=[RSP] + "Profile.TPA_Identity.@DomainID" [DOMAINID]
```

An example of the syntax to determine if the response message contains a particular node is as follows (RSP has been previously set to "EPS:rsp.ProfileSearchRQ; NOT_EMPTY has been defined as Any 10 characters):

```
READ F=[RSP] + "hasNode(\"Profile\")" [NOT_EMPTY]
```

The value of [NOT_EMPTY] is "TRUE" if the response message does contain the Profile node.

(Note that in the above example, the special Non-Print Character combination of a backslash and a quotation mark (\") has been used in place of a quotation mark (") around Profile in order to force Sabre Scribe to interpret the quotation marks as part of the character string initiated with hasNode.) See the Sabre Scribe Scripting Reference Guide for more information on Non-Print Characters.

You would use the technique described above to determine whether a node exists in the response message when the node does not contain a value. For example, this is commonly the case for the Success node.

An example from the emulator.log file of a response message containing the Success node is shown below:

```
<Sabre_OTA_ProfileCreateRS xmlns="http://www.sabre.com/eps/schemas" TimeStamp="2009-12-10T20:14:32.836"
Version="3.2"  CreateDateTime="2009-12-10T20:14:32.836">

       <ResponseMessage>

              <Success/>

       </ResponseMessage>

       <Profile ClientCode="TN" ClientContextCode="MYS" UniqueID="102224557" ProfileTypeCode="TVL"
ProfileName="SRDUBE10DEC3" DomainID="A5CE"    ProfileStatusCode="AC"/>

</Sabre_OTA_ProfileCreateRS>
```

The following syntax would fail to capture any value in the variable [MSG] since, as can be seen in the emulator.log, there is no value in the Success node.

```
READ F=[RSP] + "ResponseMessage.Success" [MSG]
```

However, you could determine the success of the request by determining whether the success node is present in the response as follows:

```
READ F=[RSP] + "ResponseMessage.hasNode(\"Success\")" [NOT_EMPTY]
```

When the Success node is present, the value of [NOT_EMPTY] is "TRUE".

An example of the web service Sabre_OTA_ProfileReadRS in XML format may appear as follows:

```
Sabre_OTA_ProfileReadRS xmlns="http://www.sabre.com/eps/schemas" Version="3.0">

      <ResponseMessage><Success/></ResponseMessage>

      <Profile CreateDateTime="2009-08-11T20:02:07.262" UpdateDateTime="2009-08-

       12T16:41:17.247" PrimaryLanguageIDCode="EN-US">

            <TPA_Identity ClientCode="TN" ClientContextCode="MYS"

             UniqueID="101221000" ProfileTypeCode="TVL" ProfileName="SRSMITH"

             ProfileNameModifyIndicator="Y" DomainID="A5CE"

             ProfileStatusCode="AC"/>

            <Traveler>

                  <Customer>

                        <PersonName OrderSequenceNo="1">

                              <NamePrefix>MR</NamePrefix>

                              <GivenName>STEVEN</GivenName>

                              <MiddleName>ROBERT</MiddleName>

                              <SurName>SMITH</SurName>

                              <NameSuffix>ESQ</NameSuffix>

                        </PersonName>

                  </Customer>

                  <TPA_Extensions>

                        <PriorityRemarks Text="TEST" OrderSequenceNo="1"/>

                        <AssociatedFilters FilterID="18690" FilterName="business"

                         ClientCode="TN" ClientContextCode="MYS" DomainID="A5CE"

                         OrderSequenceNo="1"/>

                        <AssociatedFilters FilterID="18692" FilterName="leisure"

                         ClientCode="TN" ClientContextCode="MYS" DomainID="A5CE"

                         OrderSequenceNo="2"/>

                        <AssociatedTemplate TemplateID="8866" TemplateName="TRAVELER1" ClientCode="TN"
                         ClientContextCode="GLB" DomainID="A5CE"/>

                  </TPA_Extensions>

            </Traveler>

      </Profile>

</Sabre_OTA_ProfileReadRS>
```

An example of the code used to read the PersonName information in a response message received in reply to a Sabre_OTA_ProfileReadRQ is as follows:

```
READ F=[RSP] + "Profile.Traveler.Customer.PersonName.NamePrefix" [NAME_PREFIX]

READ F=[RSP] + "Profile.Traveler.Customer.PersonName.GivenName" [GIVEN_NAME]

READ F=[RSP] + "Profile.Traveler.Customer.PersonName.MiddleName" [MIDDLE_NAME]

READ F=[RSP] + "Profile.Traveler.Customer.PersonName.SurName" [SUR_NAME]

READ F=[RSP] + "Profile.Traveler.Customer.PersonName.NameSuffix" [NAME_SUFFIX]
```

Reading Multiple Child Nodes

An XML response message might contain multiple sibling nodes with the same name. For example, a profile may have several associated filters. The response to a Sabre_OTA_ProfileReadRQ could contain multiple associated filter nodes within the TPA_Extensions parent node.

```
<TPA_Extensions>

      <AssociatedFilters FilterID="18690" FilterName="business" ClientCode="TN"
       ClientContextCode="MYS" DomainID="A5CE" OrderSequenceNo="1"/>

      <AssociatedFilters FilterID="18692" FilterName="leisure" ClientCode="TN"
       ClientContextCode="MYS" DomainID="A5CE" OrderSequenceNo="2"/>

      <AssociatedTemplate TemplateID="8866" TemplateName="TRAVELER1" ClientCode="TN"
       ClientContextCode="GLB" DomainID="A5CE"/>

</TPA_Extensions>
```

The Sabre Scribe syntax to read the filters may appear as follows (assume the maximum possible number of filters is 2):

```
[I]=0
REPEAT
 READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@OrderSequenceNo"
[ORDER]
 IF [ORDER]=2 THEN
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@FilterID" [F2I]
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@FilterName" [F2N]
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@DomainID" [F2D]
 ELSEIF [FILTERORDER]=1 THEN
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@FilterID" [F1I]
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@FilterName" [F1N]
  READ F=[RSP]+"Profile.Traveler.TPA_Extensions.AssociatedFilters["+[I]+"].@DomainID" [F1D]
 ENDIF
 [I]=[I]+1
UNTIL [I]=2
```

Exercise 3

An XML response to a profile read request is shown in Appendix E. Write the scribe code to read the four phone numbers in the response message (ProfileReadRS) and present them in a script window.

Answers to the exercises are found in Appendix F of this guide.

Sample Script

A complete example of a script that reads a profile is included below.

```
DEFINE [ERR=ANY:10:OPT:::]

DEFINE [GNAME=ANY:20:OPT:::]

DEFINE [MNAME=ANY:20:OPT:::]

DEFINE [SNAME=ANY:20:OPT:::]

DEFINE [NAMESUF=ANY:5:OPT:::]

DEFINE [EMAIL=ANY:30:OPT:::]

DEFINE [EORDER=NUM:2:OPT:::]

DEFINE [EMAIL1=ANY:30:OPT:::]

DEFINE [EMAIL2=ANY:30:OPT:::]

DEFINE [E1TYPE=ANY:8:OPT:::]

DEFINE [E2TYPE=ANY:8:OPT:::]

DEFINE [TORDER=NUM:2:OPT:::]

DEFINE [TEL1=ANY:15:OPT:::]

DEFINE [T1TYPE=ANY:4:OPT:::]

DEFINE [T1TEXT=ANY:15:OPT:::]

DEFINE [TEL2=ANY:15:OPT:::]

DEFINE [T2TYPE=ANY:4:OPT:::]

DEFINE [T2TEXT=ANY:15:OPT:::]

DEFINE [REQ=ANY:50:OPT:::]

DEFINE [RSP=ANY:50:OPT:::]

DEFINE [NOT_EMPTY=ANY:10:OPT:::]

DEFINE [UNIQUE_ID=ANY:30:OPT:::]

DEFINE [I=NUM:2:OPT:::]

[REQ] = "EPS:req.ProfileReadRQ."

[RSP] = "EPS:rsp.ProfileReadRS."


WINDOW H-"Profile ID"

  Profile ID: [UNIQUE_ID]

ENDWINDOW
```

```
;Initialize the request object
WRITE F=[REQ] + "@Version" "2.2"
WRITE F=[REQ] + "@TimeStamp" "2009-06-29T09:15:37.955Z"
WRITE F=[REQ] + "Profile.TPA_Identity.@DomainID" "A5CE"
WRITE F=[REQ] + "Profile.TPA_Identity.@ProfileTypeCode" "TVL"
WRITE F=[REQ] + "Profile.TPA_Identity.@UniqueID" [PROFILE_ID]
WRITE F=[REQ] + "Profile.TPA_Identity.@ClientContextCode" "MYS"
WRITE F=[REQ] + "Profile.TPA_Identity.@ClientCode" "TN"
WRITE F=[REQ] + "Profile.IgnoreReadSubjectAreas.SubjectAreaName" "STARData"

;Send the request
»{CLEAR}Sending ProfileReadRQ...{RESET}«
OPEN F="EPS:"

;Check if request was successful
READ F="EPS:rsp.error" [ERR]
IF [ERR] = "TRUE" THEN
  WINDOW H="Error Received"
  An error was received in the response
  to the ProfileReadRQ.
  Check emulator.log
  ENDWINDOW
  EXIT
ENDIF
;Process the response
»Processing ProfileReadRS:: {RESET}«
READ F=[RSP]+"hasNode(\"Profile\")" [NOT_EMPTY] ;check if Profile node exists

IF [NOT_EMPTY]<>"TRUE" THEN ;N0 profile was found – warn and exit script
  WINDOW H="No profile found"
  No profile exists to match entered

  data. Check your entered data.

  ENDWINDOW

  EXIT

ENDIF

 ;READ NAME INFO

READ F=[RSP]+"Profile.Traveler.Customer.PersonName.GivenName" [GNAME]

READ F=[RSP]+"Profile.Traveler.Customer.PersonName.MiddleName" [MNAME]

READ F=[RSP]+"Profile.Traveler.Customer.PersonName.SurName" [SNAME]

READ F=[RSP]+"Profile.Traveler.Customer.PersonName.NameSuffix" [NAMESUF]

;READ TELEPHONES

[I]=0

REPEAT

  READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].@OrderSequenceNo" [TORDER]

  IF [TORDER]=2 THEN

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [T2TYPE]

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [T2TEXT]

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL2]

  ELSEIF [TORDER]=1 THEN

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [T1TYPE]

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [T1TEXT]

    READ F=[RSP]+"Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL1]

  ENDIF

  [I]=[I]+1

UNTIL [I]=2


[I]=0

REPEAT

  READ F=[RSP]+"Profile.Traveler.Customer.Email["+[I]+"].@OrderSequenceNo" [EORDER]

  IF [EORDER]=2 THEN

    READ F=[RSP]+"Profile.Traveler.Customer.Email["+[I]+"].@EmailTypeCode" [E2TYPE]
    READ F=[RSP]+"Profile.Traveler.Customer.Email["+[I]+"].@EmailAddress" [EMAIL2]
  ELSEIF [EORDER]=1 THEN
    READ F=[RSP]+"Profile.Traveler.Customer.Email["+[I]+"].@EmailTypeCode" [E1TYPE]
    READ F=[RSP]+"Profile.Traveler.Customer.Email["+[I]+"].@EmailAddress" [EMAIL1]
  ENDIF
  [I]=[I]+1
UNTIL [I]=2
DEFAULT [GNAME]=[GNAME] DEFAULT [MNAME]=[MNAME] DEFAULT [SNAME]=[SNAME]
DEFAULT [NAMESUF]=[NAMESUF]
DEFAULT [TEL1]=[TEL1] DEFAULT [T1TYPE]=[T1TYPE] DEFAULT [T1TEXT]=[T1TEXT]
DEFAULT [TEL2]=[TEL2] DEFAULT [T2TYPE]=[T2TYPE] DEFAULT [T2TEXT]=[T2TEXT]
DEFAULT [E2TYPE]=[E2TYPE] DEFAULT [EMAIL2]=[EMAIL2]
DEFAULT [E1TYPE]=[E1TYPE] DEFAULT [EMAIL1]=[EMAIL1]
WINDOW H="Profile Data"
 Traveler Name:
  [GNAME] [MNAME] [SNAME] [NAMESUF]
 Telephones:
  [TEL1] [T1TYPE] [T1TEXT]
  [TEL2] [T2TYPE] [T2TEXT]
 Emails:
  [EMAIL1] [E1TYPE]
  [EMAIL2] [E2TYPE]
ENDWINDOW
```

Close the Connection (Reset the Request)

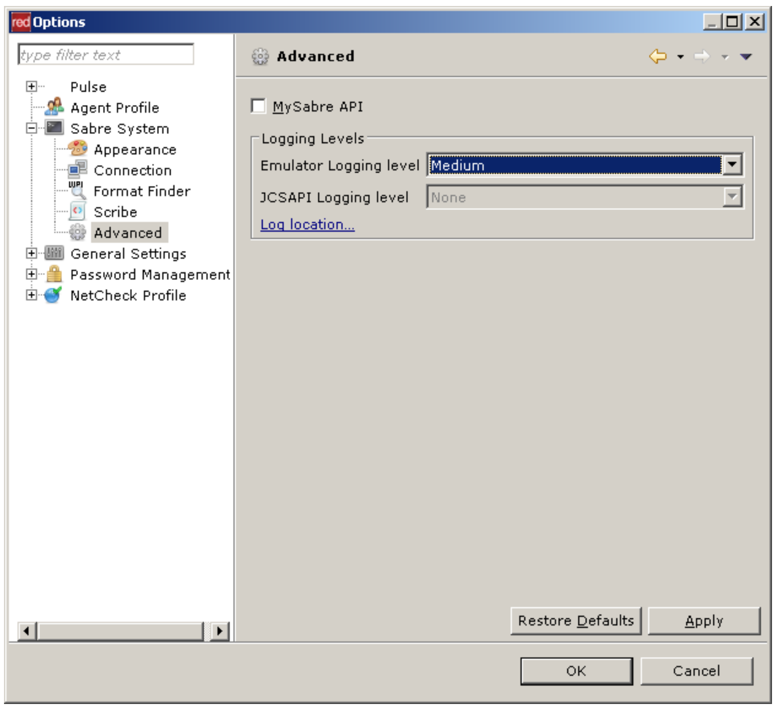
The "CLOSE" method can be used to reset the request object (in other words, remove all its child nodes and attributes):

```
CLOSE F="EPS:"
```

If you use SPAWN or CHAIN to initiate another script the request object is automatically reset in the new script.

It is possible to see the actual XML request messages created as a result of your code as well as the XML response messages received in response to the requests. To do so, set the Emulator Logging level setting in Sabre Red Workspace to low and then run the script (see Sabre Red Workspace screenshots below).

In Sabre Red Workspace go to Tools>>Options>>Advanced



Emulator Logging level: Medium
JCSAPI Logging level: None
Log location...

After running the script review the emulator.log file found in the following directory:

C:\Sabre\Apps\Emulator\Users\{*end-user's Sabre Agent ID*}\emulator.log.

The web service request and response messages will appear in the emulator.log prefaced by `WebServiceConnection`.

An example of an emulator.log file with the web services messages highlighted appears in Appendix D.

Limitations

1. Length of text fields is limited to the max length of a Scribe variable (currently 80 characters).

Should the length of the character string in a text field exceed 80 characters simply break the character string into pieces of less than 80 characters and join the parts with a plus sign as in the example below where the last two characters of the attribute name OrderSequenceNo are enclosed in separate quotations and added to the preceding string.

```
WRITE
F=[REQ]+"ProfileInfo.Profile.Traveler.TPA_Extensions.CustomerRefe
renceInfo.@OrderSequence"+"No" "1"
```

2. Sabre Scribe is only capable of consuming the 6 Sabre Profiles web services listed in this document, not all the Sabre Profiles web services that exist; therefore, should you wish to create an interface capable of consuming all Sabre Profiles web services, it is necessary to build an interface which communicates to the Sabre Profiles database directly using web services messages.

Obtaining Assistance

When seeking assistance with Sabre Scribe access to Sabre Profiles, be sure to direct your inquiry to the proper help desk depending upon the focus of your question.

If you require assistance regarding the Sabre Profiles web services, contact the Sabre Developer Resource Support Desk. For example, a question such as "What is the purpose of the ProfileNameOnly attribute of the ProfileSearchCriteria node in the ProfileSearchRQ message?" would go to the Developer Resource Support Desk. Contact information for the Developer Resource Support Desk is found on the Web Services Community page of Agency eServices.

If you require assistance regarding the syntax used within a script to access Sabre Profiles, contact the Applications Support desk at the Sabre Software Helpdesk. For example, a question such as "How do I include the DomainID attribute in the TPA_Identity node of the ProfileSearchRQ message?" would go to the Applications Support desk at Sabre Software Helpdesk.

Use of the Applications Support desk for assistance with custom Sabre Scribe projects is subject to the terms and conditions of your Support Service election as submitted by your agency on the *Sabre Scribe Developer Agency Addendum.*

Appendix A – Sample of sessionconfig section of emulator.log

The sample below represents the SessionConfig section of the emulator.log. The emulator.log file is created whenever a new session of Sabre Red Workspace is opened.

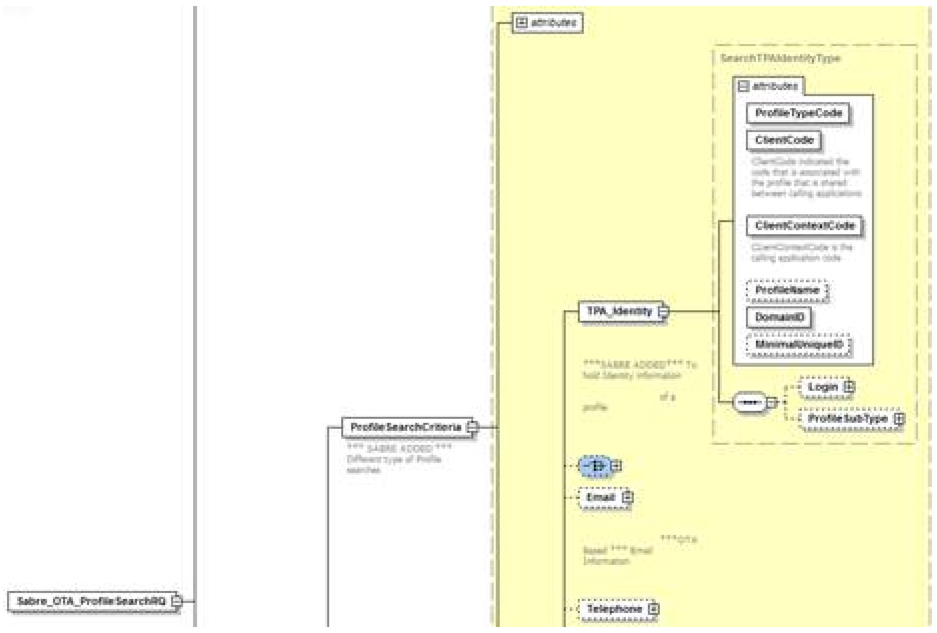
In Sabre Red Workspace check the location of the emulator.log file by going to Tools>>Options>>Advanced and clicking on the Log Location link.

Bold font is shown in the below example to highlight the list of keywords; however the text will not appear in bold font in the actual emulator.log file.

```
13:21:21.765 01 SessionConfig : createSession(): settings= {enableBSI=true,
api.license.url=https://utilservlets.cert.sabre.com/utilServlets/APILicense,
SHA59_AffiliateHost=site59.sabre.com, timeFormat=24, browser.traveltools=<html><div style="font-size:
13pt; background-color: #407ecb; font-weight: bold; color: #fffff; text-color: white; padding: 0px 5px;
text-decoration: none;"><font color="white">Travel Products</font></div></html>, log.level=0,
enableEps=true, gsm.airlinePartition=null, formatFinderIntegrateOldFlow=true,
scribe.UseStringDelimiters=false,
UETUsage.logging.url=https://utilServlets.sabre.com/utilServlets/ClientLoggingServlet,
enableAirGUI=false,
scriptlets=Air,Hotel,Car,Seat,UnusedETickets,Eps,PNRServices,Promo,PointClickParsing,EmuMarkup,Aggregator
,AirGUI,FormatFinder,AirCanada,ScribeListener, SHA59_AffiliateId=10422,
encodeDecodeServlet.server=https://utilservlets.cert.sabre.com, releaseVersion=release_13_0_009,
deploySSLVPN2Url=https://sabrevpn.cert.sabre.com/vpnclient/,
athid=Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-
XXXXXXXXXXXXXXX4686!895122!0!, HM4_Host=https://ads.virtuallythere.com/html.ng/, enableAgent59=false,
enableMidWest=true, api.version=02.00.01.0081_2330757096, ybUsage.logging.sizeLimit=30, keywords=SUBMGR
SUBAAA CCVIEW SUBACC ACCESS MULSET USGSessionUser TravelPolicyExternalUser RolesReadOnlyExternalUser
MySabreWCScribeUser EPSExternalUser TravelPolicyUser PremiumEliotUser PnrMoveUser LibertyUser
LibertyTaxDisplay LibertyPromospots LibertyPremiumSeats LibertyJR LibertyAvailFareQuote GSMUser EPSUser
EPSRolesReadOnlyUser,
```

Appendix B – ProfileSearchRQ schema viewed with third-party XML schema reader

Below is a screen shot of the ProfileSearchRQ schema as displayed in a third-party XML reading software.



Appendix C – Sample of Complete Sabre_OTA_ProfileCreateRQ and Sabre Scribe Code to Create it

Shown below is a sample of a complete Sabre_OTA_ProfileCreateRQ web service request XML as captured from the emulator.log file (Part 1). This is followed by an example of only the portion of the message that must be created within the script. Following that is an example of the actual Sabre Scribe code that would result in the creation of the request message seen below (Part 2). Sample XML such as this may be found on at the Developer Resource Center.

Part 1:

```
WebServiceConnection : requestXML= <?xml version="1.0" encoding="UTF-8"?><soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0" soap-env:mustUnderstand="1"><eb:From><eb:PartyId eb:type="URI">Scribe</eb:PartyId></eb:From><eb:To><eb:PartyId eb:type="URI">https://webservices.sabre.com/websvc</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:ConversationId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!ICESMSLB\/RES.LB!-XXXXXXXXXXXXXXX2925!854537!0!606E0B4A82218F</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EPS_ProfileCreateRQ</eb:Action><eb:MessageData><eb:MessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!ICESMSLB\/RES.LB!-XXXXXXXXXXXXXXX2925!854537!0!606E0B4A82218F</eb:MessageId><eb:Timestamp>2009-09-14T20:47:48</eb:Timestamp></eb:MessageData></eb:MessageHeader><wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String" EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!ICESMSLB\/RES.LB!-XXXXXXXXXXXXXXX2925!854537!0!606E0B4A82218F</wsse:BinarySecurityToken></wsse:Security></soap-env:Header><soap-env:Body>

<Sabre_OTA_ProfileCreateRQ Version="1.0" xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemasWSDL\Sabre_OTA_ProfileCreateRQ.xsd" xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >


<Profile UpdateDateTime="2009-07-14T07:42:10.727Z" CreateDateTime="2009-07-14T07:42:10.727Z" ><TPA_Identity ClientContextCode="MYS" ClientCode="TN" ProfileName="SRDUBE14SEP3" DomainID="A5BE" ProfileTypeCode="TVL" ProfileStatusCode="AC" UniqueID="*" ></TPA_Identity><Traveler><Customer><PersonName OrderSequenceNo="1" ><GivenName>STEVEN</GivenName><SurName>DUBE</SurName></PersonName><Telephone LocationTypeCode="AGY" OrderSequenceNo="1" ><FullPhoneNumber>XXXXXXXXXX2420</FullPhoneNumber></Telephone></Customer><TPA_Extensions><AssociatedTemplate ClientContextCode="MYS" ClientCode="TN" DomainID="A5BE" TemplateID="3595" TemplateName="srdube14augPRODa5be" ></AssociatedTemplate></TPA_Extensions></Traveler></Profile></Sabre_OTA_ProfileCreateRQ></soap-env:Body></soap-env:Envelope>

15:47:49.219 10 XMLParser : parseElements () : <?xml version="1.0" encoding="UTF-8"?>
```

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-
env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0"
soap-env:mustUnderstand="1"><eb:From><eb:PartyId
eb:type="URI">https://webservices.sabre.com/websvc</eb:PartyId></eb:From><eb:To><eb:PartyId
eb:type="URI">Scribe</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:ConversationId>Shared/IDL:IceSess\/
SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!ICESMSLB\/RES.LB!-
XXXXXXXXXXXXXXX2925!854537!0!606E0B4A82218F</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EP
S_ProfileCreateRS</eb:Action><eb:MessageData><eb:MessageId>402bddc2-232c-4730-89cd-
d295e377fbcc@144</eb:MessageId><eb:Timestamp>2009-09-
14T20:47:49</eb:Timestamp><eb:RefToMessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!IC
ESMSLB\/RES.LB!-
XXXXXXXXXXXXXXX2925!854537!0!606E0B4A82218F</eb:RefToMessageId></eb:MessageData></eb:MessageHeader><wsse:
Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken
valueType="String"
EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESB!ICESMSLB\/RES.
LB!-XXXXXXXXXXXXXXX2925!854537!0</wsse:BinarySecurityToken></wsse:Security></soap-env:Header><soap-
env:Body><Sabre_OTA_ProfileCreateRS xmlns="http://www.sabre.com/eps/schemas" TimeStamp="2009-09-
14T20:47:49.172" Version="2.2"><ResponseMessage><Success/></ResponseMessage><Profile ClientCode="TN"
ClientContextCode="MYS" UniqueID="102219823" ProfileTypeCode="TVL" ProfileName="SRDUBE14SEP3"
DomainID="A5BE" ProfileStatusCode="AC"/></Sabre_OTA_ProfileCreateRS></soap-env:Body></soap-env:Envelope>


15:47:49.219 10 >>>> XMLParser.startDocument()
```

The portion of the web service request that must be created within the script is only the portion within the <soap-env:Body> and </soap-env:Body> tags. The remainder of the message is created automatically by the Sabre Scribe runtime.

The portion of request that must be created in the script is shown again below but formatted so that the XML tags which will be translated into the nodes of the script syntax are easy to see.

```
<Sabre_OTA_ProfileCreateRQ Version="1.0"
xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemasWSDL\Sabre_OTA_ProfileCreateRQ.xsd"
xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >


        <Profile UpdateDateTime="2009-07-14T07:42:10.727Z" CreateDateTime="2009-07-14T07:42:10.727Z" >

                <TPA_Identity ClientContextCode="MYS" ClientCode="TN"
                   ProfileName="SRDUBE14SEP3" DomainID="A5BE" ProfileTypeCode="TVL"
                         ProfileStatusCode="AC" UniqueID="*" >

                </TPA_Identity>
                <Traveler>

                        <Customer>

                                <PersonName OrderSequenceNo="1" >

                                        <GivenName>STEVEN</GivenName>

                                        <SurName>DUBE</SurName>

                                </PersonName>

                        <Telephone LocationTypeCode="AGY" OrderSequenceNo="1" >

                                <FullPhoneNumber>860-123-1234</FullPhoneNumber>

                        </Telephone>

                        </Customer>

                        <TPA_Extensions>

                                <AssociatedTemplate ClientContextCode="MYS" ClientCode="TN"
                                        DomainID="A5BE" TemplateID="1870"
                                        TemplateName="TRAVELER1" >

                                </AssociatedTemplate>

                        </TPA_Extensions>

                </Traveler>

        </Profile>

</Sabre_OTA_ProfileCreateRQ>
```

Part 2:

This section shows the Sabre Scribe code that would create the XML message in Part 1.

```
WRITE F=[REQ] + "@Version" "2.2"

WRITE F=[REQ] + "Profile.@UpdateDateTime" "2009-07-14T07:42:10.727Z"

WRITE F=[REQ] + "Profile.@CreateDateTime" "2009-07-14T07:42:10.727Z"


WRITE F=[REQ] + "Profile.TPA_Identity.@ProfileStatusCode" "AC"

WRITE F=[REQ] + "Profile.TPA_Identity.@DomainID" "A5BE"

WRITE F=[REQ] + "Profile.TPA_Identity.@ProfileName" "SRDUBE14SEP3"

WRITE F=[REQ] + "Profile.TPA_Identity.@ProfileTypeCode" "TVL"

WRITE F=[REQ] + "Profile.TPA_Identity.@UniqueID" "*"
WRITE F=[REQ] + "Profile.TPA_Identity.@ClientContextCode" "MYS"

WRITE F=[REQ] + "Profile.TPA_Identity.@ClientCode" "TN"


WRITE F=[REQ] + "Profile.Traveler.Customer.PersonName.@OrderSequenceNo" "1"

WRITE F=[REQ] + "Profile.Traveler.Customer.PersonName.GivenName" "STEVEN"

WRITE F=[REQ] + "Profile.Traveler.Customer.PersonName.SurName" "DUBE"

WRITE F=[REQ] + "Profile.Traveler.Customer.Telephone.@OrderSequenceNo" "1"

WRITE F=[REQ] + "Profile.Traveler.Customer.Telephone.@LocationTypeCode" "AGY"

WRITE F=[REQ] + "Profile.Traveler.Customer.Telephone.FullPhoneNumber" "860-123-1234"


WRITE F=[REQ] + "Profile.Traveler.TPA_Extensions.AssociatedTemplate.@DomainID" "A5BE"

WRITE F=[REQ] + "Profile.Traveler.TPA_Extensions.AssociatedTemplate.@ClientContextCode" "MYS"

WRITE F=[REQ] + "Profile.Traveler.TPA_Extensions.AssociatedTemplate.@ClientCode" "TN"

WRITE F=[REQ] + "Profile.Traveler.TPA_Extensions.AssociatedTemplate.@TemplateName"
"TRAVELER1"

WRITE F=[REQ] + "Profile.Traveler.TPA_Extensions.AssociatedTemplate.@TemplateID" "1870"
```

Note that the XML tag structure is replicated and preserved by the node structure of the Sabre Scribe Syntax.

For example, looking at the XML sample, we see that the data for the traveler given name is housed within the XML open tag <GivenName> and close tag </GivenName> which in turn are housed within the tags <PersonName> and </PersonName> which in turn are housed within <Customer> and </Customer> which in turn are housed within <Traveler> and </Traveler> which in turn are housed within <Profile> and <Profile>.

The equivalent Sabre Scribe syntax is therefore,

```
WRITE F=[REQ] + "Profile.Traveler.Customer.PersonName.GivenName" "STEVEN"
```

Because the GivenName element has its own set of XML tags, the nodes in the Sabre Scribe syntax are separated by a period (.).

If the data element is an attribute of a set of XML tags, as in the case of the OrderSequenceNo of the PersonName XML tags, then the attribute value is prefaced by a period and at symbol (.@) as seen here:

```
WRITE F=[REQ] + "Profile.Traveler.Customer.PersonName.@OrderSequenceNo" "1"
```

Appendix D – ProfileReadRQ and ProfileReadRS from emulator.log

Below is an excerpt from an emulator.log file. The emulator logging level in the Sabre point of sale was set to high prior to running a script which involved several web service request messages and their respective responses. The areas of the emulator.log file that will be of interest in troubleshooting a script are those highlighted in blue text. The entire web service message can be seen but the main area of interest is primarily that between the tags which indicate the service request type or response in the soap envelope body (in bold text in the below example.)

```
22:42:33.862 38 Interpreter : MAIN ip= 119  oc= 36 _WRITE

22:42:33.862 38

22:42:33.862 38 IOObject : ***** new IOObject constructed, file=
EPS:req.ProfileReadRQ.Filter.@ClientContextCode

22:42:33.862 38 IOObject : writeIO, fileStr= EPS:req.ProfileReadRQ.Filter.@ClientContextCode
writeString= MYS

22:42:33.862 38 Interpreter : MAIN ip= 121  oc= 105 MOV

22:42:33.862 38 Interpreter : MAIN ip= 125  oc= 29 FILE

22:42:33.862 38 Interpreter : MAIN ip= 127  oc= 38 OPEN

22:42:33.862 38

22:42:33.862 38 IOObject : ***** new IOObject constructed, file= EPS:

22:42:33.862 38 WebServiceConnection : requestXML= <?xml version="1.0" encoding="UTF-8"?><soap-
env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-
env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0"
soap-env:mustUnderstand="1"><eb:From><eb:PartyId
eb:type="URI">Scribe</eb:PartyId></eb:From><eb:To><eb:PartyId
eb:type="URI">https://cert.webservices.sabre.com/cert</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:Co
nversationId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-
XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EP
S_EXT_ProfileReadRQ</eb:Action><eb:MessageData><eb:MessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/
!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-
XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:MessageId><eb:Timestamp>2009-09-
18T03:42:33</eb:Timestamp></eb:MessageData></eb:MessageHeader><wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String"
EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/C
RT.LB!-XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</wsse:BinarySecurityToken></wsse:Security></soap-
env:Header><soap-env:Body><Sabre_OTA_ProfileReadRQ Version="1.0"
xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemasWSDL\Sabre_OTA_ProfileReadRQ.xsd"
xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
TimeStamp="2009-06-29T09:16:37.955Z" ><Filter ClientContextCode="MYS" ClientCode="TN" DomainID="A5CE"
FilterID="21378" ></Filter></Sabre_OTA_ProfileReadRQ></soap-env:Body></soap-env:Envelope>

22:42:34.190 14 HttpURLConnectionSupport : initial getContentType returned text/html

22:42:34.190 14 >>>> HttpURLConnectionSupport.getStatusLine()

22:42:34.190 14 <<<< HttpURLConnectionSupport.getStatusLine() :: HTTP/1.1 200 OK

22:42:34.190 14 HttpURLConnectionSupport : Set-Cookie header =
SMSESSION=E9f1LjyrcZbkUIS75cxeWiS2HWEhVkfzGtirpZQ1mTlzcjSodW8dHw4ulvMUj+f59BB0KSRZryOSYsXZXUx+BRV1qV4LgR+
JvTj5uMmcWlbpuJGaEI3luvxl43RYIgSKqwWEdIvYa+leopkQF7SN135sXj7cckpLvJoQom3nydyO05I9dw8ka/Ne4s7MCHidi4oFBwZG
urpG/qMECcQiTpvcnuTvUTLNCg1aDgZXRjKbyV3lS7SefPP3aMKQ/sAFpq8jo4lmPyfPetjd8iPSzuzbz0i+u9RJdNiEZkE9XIK6JfzQV
AuG6zRb2eSZa3qCqgjQBwhrPeomY01wSlFxo72oKluOeTFA1/rIncuhLnp4X3cGV04mzuq/JpqF+4HZupWzX2e0aj3nPqnYP7PGoEuIZI
G/KupBQP5NipwzmKI4I7/taKj+gni06/bqn14d/N4AuNDJbJ30eoaKo57neclPicRCMe9Tngt4CMnaFjV83AxCfd6Ywme+ytHWh2hlNP9
QuzycND5TNUUvxdLwKK8i0bi9ZYSJOarWVQxONNnaX6GrLoFshMjytAj57gVCdBCttxTj9HzIqFMirPZ7rgmjygTSCfCSvv78vECHBJI8
Wa2MaUJL2WfjnSd3wgthJDJCqkAP1nTiLKEXfjdAO7hSnu1JJrX/HwcxMQtSqS7fYuJ+VpTwYaOSaAxqA4Ff2RA8dc7Ipma94XJDBdW0g
EXCP/SdEiWopVuQVJ7BRKi2FnhrMR1ws4jooZAWNlSfTeYtreS1CCzQ8PP28GyQXi4bkPkJ4dF4tvHLoM+kE7J7Yu4mS0FBME5Q9sV3+G
jSv47GPe2WqYtMTI3AWoUClB5KSz4EpzmPkGu2J6dGZEsXP5UTbZIq+kOfR5HH12mzGaefdM0ZzwOwzUiA+MdxlAA1+0SOM0jyaDtvhwa
C0gKPftQAVQPE0IHyCIxvZI7rRyALhwrpOV/BGqj0hSjPoGDdlxGrkmoYcki503oysLrk0bqDlisdHaa0QIeWGcGGLT2r0aUZFbZzO3tO
V7s8oNUaUQTRKtaqDiAGIZg6zglldSFOUBJN2XrahBe3AxRErZUUWoFK3nifpTBriWOfc5kR9uYm1CBKfmSP/zZmKvUZhiUAAd;
path=/; domain=.sabre.com

22:42:34.190 14 ClientRemoteLogger : logImpl () : rsp =
com.sabre.stn.client.http.HttpContentResponse@e26d2e

22:42:34.190 14 ClientRemoteLogger : logImpl () : success = true, status code = 200, status line =
HTTP/1.1 200 OK

22:42:34.190 14 ClientRemoteLogger : logImpl () : text = <html>OK for app scribeScriptUsage with logger
org.apache.log4j.Logger@107f742</html>

22:42:34.190 14 WorkingThread : processTask () : after task.run () in
Thread[LoggingSupportWorkingThread,1,https://my.cert.sabre.com/jars/-threadGroup]

22:42:34.190 14 WorkingThread : run () : in while loop, done = false

22:42:34.190 14 WorkingThread : run () : begin wait shuttingDown = false

22:42:34.237 38 WebServiceConnection : responseXML= <?xml version="1.0" encoding="UTF-8"?>

<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-
env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0"
soap-env:mustUnderstand="1"><eb:From><eb:PartyId
eb:type="URI">https://cert.webservices.sabre.com/cert</eb:PartyId></eb:From><eb:To><eb:PartyId
eb:type="URI">Scribe</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:ConversationId>Shared/IDL:IceSess\/
SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-
XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EP
S_ProfileReadRS</eb:Action><eb:MessageData><eb:MessageId>9fef2781-40bc-4c4d-872c-
ba4d21916de5@45</eb:MessageId><eb:Timestamp>2009-09-
18T03:42:34</eb:Timestamp><eb:RefToMessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC
!ICESMSLB\/CRT.LB!-
XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:RefToMessageId></eb:MessageData></eb:MessageHeader><wsse:
```

Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String" EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-XXXXXXXXXXXXXXX9576!904242!0</wsse:BinarySecurityToken></wsse:Security></soap-env:Header><soap-env:Body><**Sabre_OTA_ProfileReadRS** xmlns="http://www.sabre.com/eps/schemas" Version="3.1"><ResponseMessage><Success/></ResponseMessage><Filter FilterID="21378" DomainID="A5CE" ClientCode="TN" ClientContextCode="MYS" FilterName="leisure" CreateDateTime="2009-09-18T03:41:54.779" UpdateDateTime="2009-09-18T03:41:54.779" FilterStatusCode="AC" FilterTypeCode="TVL"><Profile><TPA_Identity ClientCode="TN" ClientContextCode="MYS" UniqueID="a" ProfileTypeCode="TVL" DomainID="A5CE"/><Traveler><Customer><PersonName OrderSequenceNo="1"><NamePrefix></NamePrefix><GivenName> </GivenName><MiddleName> </MiddleName><SurName> </SurName><NameSuffix></NameSuffix></PersonName><Telephone LocationTypeCode="BUS" OrderSequenceNo="2" InformationText="XX"><FullPhoneNumber>XX</FullPhoneNumber></Telephone><Telephone LocationTypeCode="HOM" OrderSequenceNo="1" InformationText="XX"><FullPhoneNumber>XX</FullPhoneNumber></Telephone><Email EmailAddress="xyz@xyz.com" OrderSequenceNo="1" EmailTypeCode="HOM"/><Email EmailAddress="xyz@xyz.com" OrderSequenceNo="2" EmailTypeCode="BUS"/><Address LocationTypeCode="HOM" Attention="XX" OrderSequenceNo="1"><AddressLine>XX</AddressLine><AddressLine>XX</AddressLine><CityName>XX</CityName><PostalCd>XX</PostalCd><StateCode>AJ</StateCode><CountryCode>AF</CountryCode></Address><Address Attention="XX" OrderSequenceNo="1"><AddressLine>XX</AddressLine><AddressLine>XX</AddressLine><CityName>XX</CityName><PostalCd>XX</PostalCd><StateCode>AJ</StateCode><CountryCode>AF</CountryCode></Address><PaymentForm OrderSequenceNo="1" TripTypeCode="AZ"><PaymentCard BankCardVendorCode="RJ" CardNumber="00" ExpireDate="122099"/></PaymentForm></Customer></Traveler></Profile></Filter></**Sabre_OTA_ProfileReadRS**></soap-env:Body></soap-env:Envelope>

22:42:34.252 38 XMLParser : parseElements () : <?xml version="1.0" encoding="UTF-8"?>

<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0" soap-env:mustUnderstand="1"><eb:From><eb:PartyId eb:type="URI">https://cert.webservices.sabre.com/cert</eb:PartyId></eb:From><eb:To><eb:PartyId eb:type="URI">Scribe</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:ConversationId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EPS_ProfileReadRS</eb:Action><eb:MessageData><eb:MessageId>9fef2781-40bc-4c4d-872c-ba4d21916de5@45</eb:MessageId><eb:Timestamp>2009-09-18T03:42:34</eb:Timestamp><eb:RefToMessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-XXXXXXXXXXXXXXX9576!904242!0!616C013A8D06C0</eb:RefToMessageId></eb:MessageData></eb:MessageHeader><wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String" EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/ACPCRTC!ICESMSLB\/CRT.LB!-XXXXXXXXXXXXXXX9576!904242!0</wsse:BinarySecurityToken></wsse:Security></soap-env:Header><soap-env:Body><**Sabre_OTA_ProfileReadRS** xmlns="http://www.sabre.com/eps/schemas" Version="3.1"><ResponseMessage><Success/></ResponseMessage><Filter FilterID="21378" DomainID="A5CE" ClientCode="TN" ClientContextCode="MYS" FilterName="leisure" CreateDateTime="2009-09-18T03:41:54.779"

UpdateDateTime="2009-09-18T03:41:54.779" FilterStatusCode="AC"
FilterTypeCode="TVL"><Profile><TPA_Identity ClientCode="TN" ClientContextCode="MYS" UniqueID="a"
ProfileTypeCode="TVL" DomainID="A5CE"/><Traveler><Customer><PersonName OrderSequenceNo="1"><NamePrefix>
</NamePrefix><GivenName> </GivenName><MiddleName> </MiddleName><SurName> </SurName><NameSuffix>
</NameSuffix></PersonName><Telephone LocationTypeCode="BUS" OrderSequenceNo="2"
InformationText="XX"><FullPhoneNumber>XX</FullPhoneNumber></Telephone><Telephone LocationTypeCode="HOM"
OrderSequenceNo="1" InformationText="XX"><FullPhoneNumber>XX</FullPhoneNumber></Telephone><Email
EmailAddress="xyz@xyz.com" OrderSequenceNo="1" EmailTypeCode="HOM"/><Email EmailAddress="xyz@xyz.com"
OrderSequenceNo="2" EmailTypeCode="BUS"/><Address LocationTypeCode="HOM" Attention="XX"
OrderSequenceNo="1"><AddressLine>XX</AddressLine><AddressLine>XX</AddressLine><CityName>XX</CityName><Pos
talCd>XX</PostalCd><StateCode>AJ</StateCode><CountryCode>AF</CountryCode></Address><Address
Attention="XX"
OrderSequenceNo="1"><AddressLine>XX</AddressLine><AddressLine>XX</AddressLine><CityName>XX</CityName><Pos
talCd>XX</PostalCd><StateCode>AJ</StateCode><CountryCode>AF</CountryCode></Address><PaymentForm
OrderSequenceNo="1" TripTypeCode="AZ"><PaymentCard BankCardVendorCode="RJ" CardNumber="00"
ExpireDate="122099"/></PaymentForm></Customer></Traveler></Profile></Filter></**Sabre_OTA_ProfileReadRS**></s
oap-env:Body></soap-env:Envelope>

```
22:42:34.252 38 >>>> XMLParser.startDocument()

22:42:34.252 38 XMLParser : characters () : https://cert.webservices.sabre.com/cert

22:42:34.252 38 XMLElement : Element [eb:PartyId].addText (https://cert.webservices.sabre.com/cert)

22:42:34.252 38 XMLParser : characters () : Scribe

22:42:34.252 38 XMLElement : Element [eb:PartyId].addText (Scribe)

22:42:34.252 38 XMLParser : characters () : A5CE

22:42:34.252 38 XMLElement : Element [eb:CPAId].addText (A5CE)
```

Appendix E – Excerpt from emulator.log for Exercise 3

The below represents a sample ProfileReadRQ sent via a Sabre Scribe script as captured by the emulator.log file and the ProfileReadRS response received in reply. Base your answer to exercise 3 on the ProfileReadRS.

Request:

```
WebServiceConnection : requestXML= <?xml version="1.0" encoding="UTF-8"?><soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/"><soap-env:Header><eb:MessageHeader
xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0" soap-
env:mustUnderstand="1"><eb:From><eb:PartyId eb:type="URI">Scribe</eb:PartyId></eb:From><eb:To><eb:PartyId
eb:type="URI">https://webservices.sabre.com/websvc</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:Conve
rsationId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESC!ICESMSLB\/RES.LB!-
XXXXXXXXXXXXXXX8204!541554!0!698FE0E1C53E64</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EP
S_EXT_ProfileReadRQ</eb:Action><eb:MessageData><eb:MessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/
!ICESMS\/RESC!ICESMSLB\/RES.LB!-
XXXXXXXXXXXXXXX8204!541554!0!698FE0E1C53E64</eb:MessageId><eb:Timestamp>2009-10-
14T16:34:56</eb:Timestamp></eb:MessageData></eb:MessageHeader><wsse:Security
xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String"
EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESC!ICESMSLB\/RES.
LB!-XXXXXXXXXXXXXXX8204!541554!0!698FE0E1C53E64</wsse:BinarySecurityToken></wsse:Security></soap-
env:Header><soap-env:Body><Sabre_OTA_ProfileReadRQ Version="1.0"
xsi:schemaLocation="http://www.sabre.com/eps/schemas..\schemas WSDL\Sabre_OTA_ProfileReadRQ.xsd"
xmlns="http://www.sabre.com/eps/schemas" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
TimeStamp="2009-06-29T09:15:37.955Z" ><Profile><TPA_Identity ClientContextCode="MYS" ClientCode="TN"
DomainID="A5CE" ProfileTypeCode="TVL" UniqueID="102222377"
></TPA_Identity></Profile></Sabre_OTA_ProfileReadRQ></soap-env:Body></soap-env:Envelope>
```

Response:

```
11:34:56.947 35 WebServiceConnection : responseXML= <?xml version="1.0" encoding="UTF-8"?>

<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"><soap-
env:Header><eb:MessageHeader xmlns:eb="http://www.ebxml.org/namespaces/messageHeader" eb:version="1.0"
soap-env:mustUnderstand="1"><eb:From><eb:PartyId
eb:type="URI">https://webservices.sabre.com/websvc</eb:PartyId></eb:From><eb:To><eb:PartyId
eb:type="URI">Scribe</eb:PartyId></eb:To><eb:CPAId>A5CE</eb:CPAId><eb:ConversationId>Shared/IDL:IceSess\/
SessMgr:1\.0.IDL/Common/!ICESMS\/RESC!ICESMSLB\/RES.LB!-
XXXXXXXXXXXXXXX8204!541554!0!698FE0E1C53E64</eb:ConversationId><eb:Service>Test</eb:Service><eb:Action>EP
S_ProfileReadRS</eb:Action><eb:MessageData><eb:MessageId>ed90fa47-7c31-4388-9eef-
997d1349bff5@160</eb:MessageId><eb:Timestamp>2009-10-
```

14T16:34:56</eb:Timestamp><eb:RefToMessageId>Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESC!ICESMSLB\/RES.LB!-XXXXXXXXXXXXXXX8204!541554!0!698FE0E1C53E64</eb:RefToMessageId></eb:MessageData></eb:MessageHeader><wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"><wsse:BinarySecurityToken valueType="String" EncodingType="wsse:Base64Binary">Shared/IDL:IceSess\/SessMgr:1\.0.IDL/Common/!ICESMS\/RESC!ICESMSLB\/RES.LB!-XXXXXXXXXXXXXXX8204!541554!0</wsse:BinarySecurityToken></wsse:Security></soap-env:Header><soap-env:Body><Sabre_OTA_ProfileReadRS xmlns="http://www.sabre.com/eps/schemas" Version="3.1"><ResponseMessage><Success/></ResponseMessage><Profile CreateDateTime="2009-10-14T16:15:50.043" UpdateDateTime="2009-10-14T16:15:50.043" PrimaryLanguageIDCode="EN-US"><TPA_Identity ClientCode="TN" ClientContextCode="MYS" UniqueID="102222377" ProfileTypeCode="TVL" ProfileName="khan/surina" ProfileNameModifyIndicator="Y" ProfileDescription="surina khan" DomainID="A5CE" ProfileStatusCode="AC"/><Traveler><Customer><PersonName OrderSequenceNo="1"><NamePrefix>ms</NamePrefix><GivenName>surina</GivenName><MiddleName>ali</MiddleName><SurName>khan</SurName></PersonName><Telephone LocationTypeCode="CEL" OrderSequenceNo="3" InformationText="not for international travel"><FullPhoneNumber>802-333-4444</FullPhoneNumber></Telephone><Telephone LocationTypeCode="HOM" OrderSequenceNo="1" InformationText="primary residence"><FullPhoneNumber>802-111-2222</FullPhoneNumber></Telephone><Telephone LocationTypeCode="UNK" OrderSequenceNo="4" InformationText="secondary residence london"><FullPhoneNumber>44-207-499-7071</FullPhoneNumber></Telephone><Telephone LocationTypeCode="BUS" OrderSequenceNo="2" InformationText="primary business"><FullPhoneNumber>802-222-3333</FullPhoneNumber></Telephone><Email EmailTypeCode="AGY" EmailAddress="skhan1@popmail.uk" OrderSequenceNo="3"/><Email EmailTypeCode="HOM" EmailAddress="khancan@hotmail.com" OrderSequenceNo="1"/><Email EmailTypeCode="CMP" EmailAddress="khaninternational@khanassociates.com" OrderSequenceNo="4"/><Email EmailTypeCode="BUS" EmailAddress="surinakhan@khanassociates.com" OrderSequenceNo="2"/><Address LocationTypeCode="HOM" Attention="surina khan" OrderSequenceNo="1"><AddressLine>1000 e. north ave</AddressLine><AddressLine>unit 29a</AddressLine><CityName>chicago</CityName><PostalCd>43212</PostalCd><StateCode>IL</StateCode><CountryCode>US</CountryCode></Address><PaymentForm OrderSequenceNo="1" TripTypeCode="AZ"><PaymentCard BankCardVendorCode="BA" CardNumber="XXXXXXXXXXXX1111" MaskedCardNumber="1111" CCViewAccess="Y" ExpireDate="122012"><Address OrderSequenceNo="1"/></PaymentCard></PaymentForm></Customer><TPA_Extensions><PriorityRemarks Text="do not send leisure trip itins to business email" OrderSequenceNo="1"/><CustomerReferenceInfo TripTypeCode="AZ" BranchID="999" ReferenceID="8888888" OrderSequenceNo="1"/><AssociatedTemplate TemplateID="4997" TemplateName="TRAVELER2" ClientCode="TN" ClientContextCode="GLB" DomainID="A5CE"/></TPA_Extensions></Traveler></Profile></Sabre_OTA_ProfileReadRS></soap-env:Body></soap-env:Envelope>

Exercise 1 answer:

ProfileReadRQ. The ProfileReadRQ service is used to read Profiles, Templates, Formats and Filters.

Exercise 2 sample answer:

```
DEFINE [REQ=ANY:50:OPT:::]
DEFINE [FORMATID=*:10]
DEFINE [FORMATDOMAIN=*:4]

[REQ]="EPS:req.ProfileReadRQ."

WINDOW
FORMATID = [FORMATID]
FORMATDOMAIN = [FORMATDOMAIN]
ENDWINDOW

WRITE F=[REQ] + "@Version" "2.2"
WRITE F=[REQ] + "@TimeStamp" "2009-06-29T09:16:37.955Z"
WRITE F=[REQ] + "Format.@FormatID" [FORMATID]
WRITE F=[REQ] + "Format.@DomainID" [FORMATDOMAIN]
WRITE F=[REQ] + "Format.@ClientCode" "TN"
WRITE F=[REQ] + "Format.@ClientContextCode" "MYS"

OPEN F="EPS:"
```

Exercise 3 sample answer:

```
DEFINE [I=NUM::OPT:::]
DEFINE [TELORDER=N]
DEFINE [TEL1PHONE=ANY:16:OPT:::]
DEFINE [TEL2PHONE=ANY:16:OPT:::]
DEFINE [TEL3PHONE=ANY:16:OPT:::]
DEFINE [TEL4PHONE=ANY:16:OPT:::]
DEFINE [TEL1TYPE=ANY:4:OPT:::]
DEFINE [TEL2TYPE=ANY:4:OPT:::]
DEFINE [TEL3TYPE=ANY:4:OPT:::]
DEFINE [TEL4TYPE=ANY:4:OPT:::]
DEFINE [TEL1TEXT=ANY:30:OPT:::]
DEFINE [TEL2TEXT=ANY:30:OPT:::]
DEFINE [TEL3TEXT=ANY:30:OPT:::]
DEFINE [TEL4TEXT=ANY:30:OPT:::]

[I]=0
REPEAT
 READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@OrderSequenceNo" [TELORDER]

 IF [TELORDER]=4 THEN
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [TEL4TYPE]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [TEL4TEXT]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL4PHONE]
 ELSEIF [TELORDER]=3 THEN
```

```
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [TEL3TYPE]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [TEL3TEXT]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL3PHONE]
 ELSEIF [TELORDER]=2 THEN
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [TEL2TYPE]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [TEL2TEXT]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL2PHONE]
 ELSEIF [TELORDER]=1 THEN
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@LocationTypeCode" [TEL1TYPE]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].@InformationText" [TEL1TEXT]
  READ F=[RSP] + "Profile.Traveler.Customer.Telephone["+[I]+"].FullPhoneNumber" [TEL1PHONE]
 ENDIF
 [I]=[I]+1
UNTIL [I]=4
DEFAULT [TEL1PHONE]=[TEL1FULLPHONE] DEFAULT [TEL1TYPE]=[TEL1TYPE] DEFAULT [TEL1TEXT]=[TEL1TEXT]
DEFAULT [TEL2PHONE]=[TEL2FULLPHONE] DEFAULT [TEL2TYPE]=[TEL2TYPE] DEFAULT [TEL2TEXT]=[TEL2TEXT]
DEFAULT [TEL3PHONE]=[TEL3FULLPHONE] DEFAULT [TEL3TYPE]=[TEL3TYPE] DEFAULT [TEL3TEXT]=[TEL3TEXT]
DEFAULT [TEL4PHONE]=[TEL4FULLPHONE] DEFAULT [TEL4TYPE]=[TEL4TYPE] DEFAULT [TEL4TEXT]=[TEL4TEXT]
WINDOW
Telephone numbers:
1: [TEL1PHONE] [TEL1TYPE] [TEL1TEXT]
2: [TEL2PHONE] [TEL2TYPE] [TEL2TEXT]
3: [TEL3PHONE] [TEL3TYPE] [TEL3TEXT]
4: [TEL4PHONE] [TEL4TYPE] [TEL4TEXT]
ENDWINDOW
```