Oracle® Hospitality OPERA Exchange Interface

Communication Vendor Specification

October 2017

ORACLE

Contents

Preface

This document explains the communication mechanisms supported by OPERA Xchange Interface (OXI).

Audience

This document is intended for software developers.

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received and any associated log files
- Screen shots of each step you take

Documentation

Oracle Hospitality product documentation is available on the Oracle Help Center at http://docs.oracle.com/en/industries/hospitality/

1 Introduction

OXI supports the following communication mechanisms. Details of each mechanism are provided later.

- HTTP/HTTPS
- FTP
- File System

When OXI is implemented as a 2-way interface, the most fundamental operation it needs to perform is receiving data to process from an external system and sending data to the external system for processing.

2 Synchronous or Asynchronous

Even though some of these communication mechanisms such as HTTP support synchronous communication, OXI processing is always asynchronous.

When OXI receives data from an external system, that data may not be processed immediately. It is queued for processing and a synchronous response is provided to notify the status of the receipt of the message -whether it has been successfully received or an error occurred while receiving/queuing the message – following the standards defined for that particular communication mechanism, if the communication mechanism supports a synchronous response. For example, 200 OK/400 Bad Request in case of http communication. An asynchronous response conforms to the result XML schema is generated for the external system stating the completion status of the processing and any other relevant information later, once the queued data is processed.

Similarly, when OXI delivers messages to external systems, it expects only an acknowledgement of the receipt of the message in the synchronous response, if applicable. Processing status and any relevant detail should be notified as an asynchronous message that conforms to the result XML schema.

3 Push or Pull

Communication mechanisms that use the client/server terminology such as HTTP and FTP can be implemented in different configurations. Since there are two parties involved in the communication – in our case OXI and the external system – theoretically either can function as the server and the other party can function as client. OXI does not support all such combinations of configurations due to various reasons. Supported configurations, where applicable, are explained while discussing the corresponding communication mechanisms later in this document.

4 HTTP/HTTPS

OXI supports the four possible client server configurations while communicating using HTTP/HTTPS. Following details apply to both HTTP and HTTPS. To use HTTPS, server has to be configured accordingly.

Important: HTTPS functionality when OXI functions as HTTP Client was tested and verified only using Apache HTTP server 1.3 with Verisign's SSL certificate.

Configuration Options

Push/Push

Both systems push messages to other system when messages are available. External system sends messages for OXI to OPERA's HTTP server and OXI sends messages to external system's HTTP server.

Pros	Cons
No polling is involved to check the availability of messages. Reduces network activity.	Both systems have to maintain HTTP servers.

Push/Pull

- External system pushes messages to OPERA when messages are available and polls OPERA's HTTP server for availability of messages (or)
- OPERA pushes messages to external system when messages are available and polls external system's HTTP server for availability of messages.

Pros	Cons
Only one side need to have the HTTP server.	To receive messages, one system will have to poll the other system's HTTP server for available messages at frequent intervals. This increases network activity.

Pull/Pull

Both systems poll the other system for messages and pull them when available. This configuration is included for the completion of the discussion and should not be implemented as this does not have any pros and have all cons of the previous two options.

Recommendations

Either first or second option should be selected after weighing the pros and cons of both options. Third option is not recommended at all as it does not seem to have any pros and has combined cons of the other two options.

OXI as Client: Receive Data from External System

An HTTP GET operation will be performed to retrieve available messages for the property from the external system. If message is available for the property, external system should return 200 OK and should send XML message in the response stream. If there is no message available for the property, external system should set the value of Content-Length in the response header to 0 and return 200 OK/204 No Content. At the time of this writing, OXI relies on the 0 Content-Length to recognize that there is no message.

HTTP Operation : GET

URL Parameters

Notes : Parameter names will be in camel case as shown below and values will be in upper case.

Name	Description
propertyName	Resort code used by the external system to identify the property. This will be the same as external resort set in OXI's configuration.

Expected HTTP Response

Result	Status Code	Content-Type	Content-Length	Content
Message available	200 OK	text/xml (UTF-8)	not used	XML message
Message not available	204 No Content/ 200 OK	not applicable	0	None
Error	>= 400	text/plain	not used	Error message

Mandatory Processing Instruction

To parse the XML message received from the external system, OXI needs to know some basic information about the message. OXI expects a processing instruction in the XML Message as follows immediately after the XML version information, to derive this information from.

<?Label *propertyName*|*messageType*|*transactionId*|*status*?>

Tags shown in green italic above must be replaced with actual values. Details of these tags follow:

Name	Description
propertyName	Resort code used by the external system to identify the property. This should be the same as external resort set in OXI's configuration.
messageType	Type of the XML message sent by the external system. Valid values are: ALLOTMENT : Allotment INVENTORY : Inventory MESSAGEREQUEST : Message request PROFILE : Profile RATE : Rate RAVL : Rate restriction RAVR : Rate restriction (room type) RESERVATION : Reservation RESULT : Result
transactionId	For new messages generated by the external system, external system's numeric transaction identifier that uniquely identifies this particular message. It must be between 1 and 999999999 inclusive. For RESULT messages generated in response to OXI's message, message id of the original message sent by OXI – It is used to update the status of the original message sent by OXI accordingly.
status	Reserved. Must be SUCCESS always.

Sample:

<?xml version = '1.0'?>
<?Label SLBK|PROFILE|249|SUCCESS?>

Sample URL:

```
http://oxihub.ft.micros.com:5868/servlets/ORSInterface?propertyName=SLBK

http://oxihub.ft.micros.com:5868/servlets/ORSInterface      :
                                                Configurable in OXI.
SLBK                                            : Will be replaced by
OXI using actual value.
```

OXI as Client: Send Data to External System

HTTP Operation : POST
Content-type : text/xml
Character-encoding : UTF-8

URL Parameters

Notes : Parameter names will be in camel case as shown below and values will be in upper case.

Name	Description
propertyName	Resort code used by the external system to identify the property. This will be the same as external resort set in OXI's configuration.
messageType	Type of the XML message sent by OXI. Valid values are: ALLOTMENT : Allotment HURDLE : Hurdle INVENTORY : Inventory INVENTORYQUERY : Inventory request INVENTORYSUMMARY : Inventory summary MESSAGEREQUEST : Message request PACKAGES : Package PROFILE : Profile RATE : Rate RAVL : Rate restriction RAVR : Rate restriction (room type) RESERVATION : Reservation RESULT : Result RTAV : Inventory snapshot SCHEMAVERSION : XML schema version STAY : Guest stay history
transactionId	For new messages generated by OXI, OXI's numeric transaction identifier that uniquely identifies this particular message. It will be between 1 and 999999999 inclusive. For RESULT messages generated in response to external systems' message, message id of the original message sent by the external system.
status	NEW : New non-result message. SUCCESS : Only for result messages. OXI processed original message received from the external system was successfully. FAILED : Only for result messages. OXI could not process original message received from the external system. DELETE : Only for profile messages. User deleted the profile in OPERA.

Expected HTTP Response

	Status Code	000Content-Type	Content-Length	Content
Message accepted	200 OK	not applicable	not used	Not used.
Error encountered	>= 400	text/plain	not used	Error message. OXI will resend the message after a delay.

Sample URL

```
http://oxihub.ft.micros.com:5868/servlets/ORSInterface?propertyName=SLBK&messageType=PROFILE&transactionId=208&status=SUCCESS

http://oxihub.ft.micros.com:5868/servlets/ORSInterface      :
                                                Configurable in OXI.
SLBK, PROFILE, 208, SUCCESS                     : Will be replaced by
OXI using actual values.
```

OXI as Server: Send Data to OXI

HTTP Operation: POST
Content-type: text/xml
Character-encoding: UTF-8
URL: http://*OperaHTTPServer*:*Port*/Operajserv/OXIServlets/PMSInterface

All the components after the port in the URL are case-sensitive. They should be used in the same case as shown above.

Implementation Change

Applicable to only those OPERA and OXI versions prior to 3.0.1 used the PL/SQL Gateway component for Apache provided by Oracle to facilitate the http communication. This mandated the clients to use form-encoded data streams while sending data to OXI's communication http server. Also it had issues handling https protocol and there are message size limitations. Though this is still supported, it has been deprecated and being replaced by the servlet based communication module which does not accept form-encoded data, but text/xml plain text data. At the time of this writing, PL/SQL Gateway based http communication is supported till version 3.6 of OPERA and OXI. It may not be supported in future versions. Servlet based communication module is supposed to resolve the following critical shortcomings: Ability to support https and the ability to support messages larger than 32000 bytes.

Authentication

From OPERA version 5.0.05.00 onwards, authentication information is needed to access the servlets for message exchange. Obtain the credentials from the site administrator and pass those using "Basic Authentication Scheme" defined in http standard, this involves passing "Authorization" request header with base64 encoded username and password as specified in the http standard.

URL Components

All italicised underlined words in the URL given above must be replaced with actual values. Description of those fields follow:

OperaHTTPServer	Replace with the IP address or host name of the OPERA HTTP server designated for communication
Port	HTTP service's listening port on the OPERA HTTP server. Optional. Specify if different from standard port 80.

URL Parameters

Notes : Parameter names and values are case sensitive. Parameter names must be in camel case as shown below and values must be in upper case.

Name	Description
interfaceName	Interface id assigned to the external system.
propertyName	Property code used by the external system to identify the property. This should be the same as the value configured in external resort field of OXI's interface setup.
messageType	Type of the XML message. Refer to OXI as Client : Receive Data from External System section for valid values
transactionId	For RESULT messages : OXI transaction id of the original message received from OXI. For other messages : External system's unique transaction identifier. Can be alphanumeric up to 100 characters long. To make it user and support friendly, it is recommended to use numeric values up to 9 digits long.

status	NEW	:	New non-result message.
	SUCCESS	:	Only for result messages. To indicate successful processing of original message received from OXIHUB by the external system.
	FAILED	:	Only for result messages. To indicate unsuccessful processing of original message received from OXIHUB by the external system.

HTTP Response

Result	Status Code	Content-Type	Content
Message accepted	200 OK	text/plain	Brief status message.
Invalid request from client	400 Bad Request	text/plain	Error message
Authentication failure	401 Unauthorized	text/html	Generic error indicating an issue with authorization.
Authentication failure	403 Forbidden	text/plain	Generic error indicating an issue with authorization.
Error reported by OXI while processing request	500 Internal Server Error	text/plain	Error message

Sample URL

http://oxihub.ft.micros.com:5868/Operajserv/OXIServlets/PMSInterface?interfaceName=ORS&propertyName=MEXICO&messageType=PROFILE&transactionId=60528&status=NEW

Sample POST Data

Content-type : text/xml

Content

```
<?xml version = '1.0'?>
<?Label MEXICO|PROFILE|60528|NEW?>
<Profile profileType="CORPORATE" gender="UNKNOWN"
xmlns="profile.fidelio.2.0">
   <profileID>1111111</profileID>
   <creatorCode>Creator</creatorCode>
   <createdDate>2004-06-09T15:29:48.000</createdDate>
   <lastUpdaterCode>Creator</lastUpdaterCode>
```

```
   <lastUpdated>2004-06-09T15:30:22.000</lastUpdated>
   <genericName>Master Account</genericName>
   <IndividualName>
      <nameSur>Master Account</nameSur>
   </IndividualName>
   <PostalAddresses>
      <PostalAddress addressType="BUSINESS">
         <address1>1111 Street</address1>
         <city>Anytown</city>
         <stateCode>FL</stateCode>
         <postalCode>99999</postalCode>
         <countryCode>US</countryCode>
         <mfPrimaryYN>Y</mfPrimaryYN>
      </PostalAddress>
   </PostalAddresses>
   <PhoneNumbers>
      <PhoneNumber phoneNumberType="BUSINESS">
         <phoneNumber>123-456-7890</phoneNumber>
         <mfPrimaryYN>Y</mfPrimaryYN>
      </PhoneNumber>
   </PhoneNumbers>
   <mfResort>MEXICO</mfResort>
   <mfAllowMail>NO</mfAllowMail>
   <mfAllowEMail>NO</mfAllowEMail>
   <mfGuestPriv>NO</mfGuestPriv>
   <mfAllowPhone>0</mfAllowPhone>
   <mfAllowSMS>0</mfAllowSMS>
</Profile>
```

OXI as Server: Receive Data from OXI

Since OXI can interface with multiple interfaces for multiple properties, external systems must identify themselves and specify the property for which they are querying in the URL parameters.

HTTP Operation: GET
Content-type: text/xml
Character-encoding: UTF-8
URL: http://*OperaHTTPServer*:*Port*/Operajserv/OXIServlets/PMSInterface

URL Components

All italicised underlined words in the URL given above must be replaced with actual values. Description of those fields follow:

OperaHTTPServer	Replace with the IP address or host name of the OPERA HTTP server designated for communication
Port	HTTP service's listening port on the OPERA HTTP server. Optional. Specify if different from standard port 80.

URL Parameters

Notes : Parameter names and values are case sensitive. Parameter names must be in camel case as shown below and values must be in upper case.

Name	Description
interfaceName	Interface ID assigned to the external system.
propertyName	Property code used by the external system to identify the property. This should be the same as the value configured in external resort field of OXI's interface setup.

Sample URL

URL:
http://nploxidb/Operajserv/OXIServlets/PMSInterface?interfaceName=ORS&propertyName=SLB
Request Method: GET

HTTP Response - Message is available for property

HTTP Return Code	**200 OK**
Response Stream	XML message
Response Header Fields	
Content-Type	text/xml
Content-Length	Not present
Title	*propertyName*|*messageType*|*transactionId*|*status* These values are similar to the parameters passed to OXIHUB while sending data to OXIHUB.

Sample Response

```
Response code    : 200
Response message : OK
```

Header Fields

```
1. Connection: [close]
2. title: [SLB|PROFILE|73|NEW]
3. cache-control: [no-control]
4. Date: [Fri, 21 May 2004 17:52:23 GMT]
5. Pragma: [no-cache]
6. Server: [Oracle HTTP Server Powered by Apache/1.3.19 (Win32)
mod_ssl/2.8.1 OpenSSL/0.9.5a mod_oprocmgr/1.0 mod_perl/1.25]
```

```
7. Content-Type: [text/xml; charset=UTF-8]
8. null: [HTTP/1.1 200 OK]
9. Transfer-Encoding: [chunked]
```

Output Data Stream

```
<?xml version = '1.0'?>
<?Label SLB|PROFILE|73|NEW?>
<Profile profileType="GUEST" gender="UNKNOWN"
xmlns="profile.fidelio.1.2">
   <profileID>1111111111</profileID>
   <creatorCode>OXI-OPERA</creatorCode>
   <createdDate>2004-05-18T18:00:27.000</createdDate>
   <lastUpdaterCode>OXI-OPERA</lastUpdaterCode>
   <lastUpdated>2004-05-18T18:00:29.000</lastUpdated>
   <genericName>Noshow</genericName>
   <IndividualName>
      <nameFirst>John</nameFirst>
      <nameSur>Noshow</nameSur>
   </IndividualName>
   <primaryLanguageID>E</primaryLanguageID>
   <PostalAddresses>
      <PostalAddress addressType="HOME">
         <countryCode>MX</countryCode>
         <mfPrimaryYN>Y</mfPrimaryYN>
      </PostalAddress>
   </PostalAddresses>
   <mfResort>SLB</mfResort>
   <mfResortProfileID>22222</mfResortProfileID>
   <mfNameCode>11111</mfNameCode>
   <mfAllowMail>NO</mfAllowMail>
   <mfAllowEMail>NO</mfAllowEMail>
   <mfGuestPriv>NO</mfGuestPriv>
</Profile>
```

HTTP Response - Message is not available for property

HTTP Return Code	200 OK
Response Stream	Not applicable
Response Header Fields	
Content-Type	Not Applicable
Content-Length	0
Title	Not present
PS	In future, System may return [204][No Content] to indicate that there is no data available.

Sample Response

```
Response code    : 200
Response message : OK
```

Header Fields

```
1. Content-Length: [0]
2. Connection: [close]
3. cache-control: [no-control]
4. Date: [Fri, 21 May 2004 17:51:30 GMT]
5. null: [HTTP/1.1 200 OK]
6. Pragma: [no-cache]
7. Server: [Oracle HTTP Server Powered by Apache/1.3.19 (Win32)
mod_ssl/2.8.1 OpenSSL/0.9.5a mod_oprocmgr/1.0 mod_perl/1.25]
8. Content-Type: [text/plain]
```

HTTP Response - Error occurred while processing the GET request

HTTP Return Code	**400 Bad Request** **500 Internal Server Error**
Response Stream	Applicable error message
Response Header Fields	
Content-Type	text/plain
Content-Length	Not Applicable
Title	Not Applicable

Sample Response

```
Response code    : 400
Response message : Bad Request
```

Header Fields

```
1. Connection: [close]
2. cache-control: [no-control]
3. Date: [Fri, 21 May 2004 18:27:39 GMT]
4. Pragma: [no-cache]
5. Server: [Oracle HTTP Server Powered by Apache/1.3.19 (Win32)
mod_ssl/2.8.1 OpenSSL/0.9.5a mod_oprocmgr/1.0 mod_perl/1.25]
6. Content-Type: [text/plain]
7. null: [HTTP/1.1 400 Bad Request]
8. Transfer-Encoding: [chunked]
```

Error Stream

```
Dequeue error for [SLB]: java.sql.SQLException: ORA-04068:
existing state of packages has been discarded
ORA-04061: existing state of package body
"OXIHUB46_D.INT_COMM_MAIN" has been invalidated
ORA-04065: not executed, altered or dropped package body
"OXIHUB46_D.INT_COMM_MAIN"
ORA-06508: PL/SQL: could not find program unit being called
ORA-06512: at "OXIHUB46_D.XMLDEQUEUECLOB2", line 22
```

```
ORA-06512: at line 1
```

5 FTP

In order to utilize data transfer via FTP protocol, a FTP server must exist that can receive PUT and GET commands from another system.

- OXI transfers data from the external system into the Inbound queue from where the OXI download processor takes over.
- OXI produces upload messages from the OPERA business events and stores these in the Outbound queue using its upload processor and an FTP transfer mechanism to send the messages to the external system.

External system has an FTP server – this is the standard handling

- The only way we are transmitting files between OXI and an external system using FTP.
- OXI does not need any additional hardware as it is functioning as the FTP client application.
- The external system has a FTP server installed.
- The external FTP server owner provides the OXI user with the hostname, username, password, and folder information for put and get.
- OXI sends its data load to the external FTP server using the PUT command. External system is responsible for processing messages delivered by OXI.
- OXI requests data from external system using the GET command. This is done in frequent, definable intervals. External system is responsible for placing available messages in the designated folder on the FTP server for OXI to download.
- The advantage is that the external vendor can potentially reuse communication parts that they have already in place.

Messages into OPERA

- Reservation is created in external system.
- OXI requests data from external system FTP server using FTP/GET.
- OXI receives the entire XML message, if available, and parses it, performs necessary conversion and writes the data to the OPERA database.
- The reservation in the OPERA database is accessible from any OPERA workstation.

Messages from OPERA

- Depending on the business event configuration, the OPERA user creates an activity in OPERA, which is captured in form of a business event.
- The OXI upload process dequeues the business event, validates the record, and applies data conversion.
- The result is stored as XML file.

- OXI sends the XML file to the external FTP server using FTP/PUT.

6 File System

In order to utilize the file transfer communication method, the external system vendor needs to establish communication to the OXI interface PC and place data into a defined directory on that PC or the surrounding network.

Messages into OPERA

- A message is created in the external system and sent to the OPERA property or data center to be stored in a defined import directory.
- The OXA C/C++ program running on the Microsoft Windows interface PC reads the XML file from the directory, does pre-validation of data elements, and writes the data into temporary XML tables.
- The OXA C/C++ program calls OXI PL/SQL code to read the data from the temporary XML tables.
- The normal OXI download processor handling begins.

Messages from OPERA

- An activity happens in OPERA and business events are created.
- The OXI upload processor processes the data and stores it in temporary XML tables.
- The OXA C/C++ program places the file into a specified export directory.
- The external system process picks up the XML message and establishes connection to the external system to send the message.

7 Requirements for the OXI Generic Installation

Hardware Requirements

- The OXI Win NT service must reside in a Microsoft Win NT or Microsoft Win 2000 workstation on the network. This workstation should not run any DOS based programs.
- The Property Interfaces PC at the property can be used to install OXI as it has similar hardware and software requirements.
- Single property thin client installations: The OPERA Application Server can also be used to install OXI.

Hotels in Thin Client have the Following Options

- Install OXI and service on the OPERA application server if the customer is a single or two-property installation.
- Install OXI on the OPERA application server and the OXI service on the IFC7W PC if the customer is single or multiple properties.
- Install OXI on the OPERA application server and the OXI service on the CRS vendor's PC, if such exists. An example would be Pegasus where the CRS vendor already has a dedicated PC. The drawback is that this PC still needs to be installed with the Oracle Client compatible with the Oracle database running OPERA.

Software Requirements for Running OXI

- A physical network connection between OXI service and external system is needed.
- The OXI Win NT service PC has to be equipped with Oracle Client 8I 1.7, OCI, Oracle Objects for OLE with a memory of 256 MB RAM.
- A synonym schema to OPERA is required for OXI.
- The OXI runtimes will reside in the same runtimes directory as OPERA. OXI will use Oracle forms installed for OPERA.
- The Java options in the OXI instance have to be active.
- The OPERA version must have minimum Build 45 patch 33.1+ using Oracle 8i patch 1.7 (check latest OPERA/OXI versions available).
- Oracle forms 6i using patch 8 (check latest OPERA/OXI versions available).

8 Tools

Tools that external system can use to validate the OXI XML schemas:

- Oracle xmlparser can be used to parse the XML message.
- The standard XML schemas are created before the W3C Specifications released, so they are not W3C compliant.
- The standard XML schemas are derived from HITIS specifications.
- The standard XML schemas are created using Microsoft SDK 3.0.
- The standard XML schemas are called XDR Schemas
  [XDR: The XML-Data Reduced (XDR) schema defines the individual elements, attributes, and relations used in the XML structure].