



OPERA WEB SUITE VERSION 5.1

PRODUCT OVERVIEW

Record of Changes 3
Audience 4
Overview 4
What Are Web Services? 4
What is Opera Web Suite? 5
What are the benefits of Opera Web Suite? 11
How Does Opera Web Suite Work? 13
Requirements and Recommendations 16
Support for Opera Web Suite 18
Learn More About Opera Web Suite 19
Appendix A: Know-How on Web Services 20

Record of Changes

Date	Version	Description	Author
18 Dec 2004	1.0.0	Initial Release	C.Dimitrov

Audience

This document is designed for use by our potential interface partners and clients who wish to develop chain specific User Interfaces. It will give our partners an overview of Opera Web Suite and enable them to assess the knowledge and approximate time-scale such an interface entails.

Overview

This overview is designed to provide a high-level introduction to Opera Web Suites (OWS), including a brief introduction to Web Services technologies, as well as a description of how Web Services can be used to interface with the Central Reservation System[®] (CRS) on Opera Reservation System[®] (ORS) platform.

OWS is designed to support data transfer between client travel applications and the Central Reservation System (CRS). OWS is a collection of Web Services that provide client applications with access to key functionality on the CRS/ORS.

What Are Web Services?

Web services are the fundamental building blocks in the move to distributed computing on the Internet. Open standards and the focus on communication and collaboration among people and applications have created an environment where Web services are becoming the platform for application integration. Applications are constructed using multiple Web services from various sources that work together regardless of where they reside or how they were implemented. Web Services represent an *evolution* in communication between applications, rather than a *revolution*. Most of the technologies behind Web Services are not new. However, these existing technologies have been combined and refined to create a common system of communication. Most importantly, a set of common standards have been assigned to these technologies so that systems using Web Services have a single, explicitly defined method for communicating with each other.

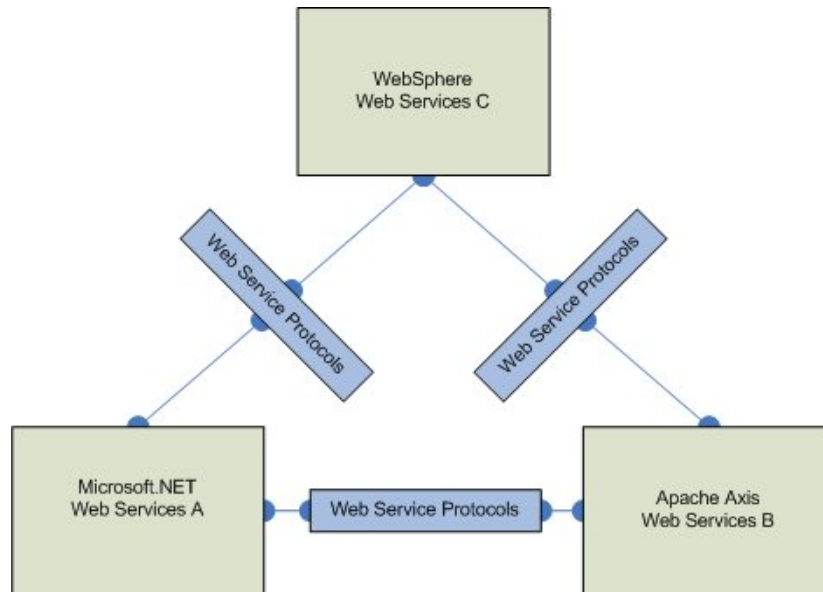
Application Interfaces before Web Services

Web Services create greater flexibility because they remove the need for applications to programmatically understand each other. In some cases, pre-existing interfaces can be purchased to translate between disparate languages and platforms. In other cases, however, a proprietary applications interface must be specifically designed and deployed to communicate with the other application. Such specialized design can often consume huge amounts of financial and employee resources, as well as add another layer of complication to product design, implementation, and maintenance.

Application Interfaces with Web Services

One of the primary advantages of the Web services architecture is that it allows programs written in different languages on different platforms to communicate with each other in a standardised way. The specific protocols for data transfer are less important than the fact that all of the services and applications adhere to the same standard. **Appendix A: Know-How on Web Services (page 19)** for more details about the specific standards, protocols, and formats used in OWS and other Web Services.

In the below scenario, we have Application A, Application B and C Web Services that both use standard Web Service protocols to transfer information.



The fact that Application A uses Microsoft.NET on a Windows platform as the infrastructure for its Web Service, while Application C uses Web Sphere on UNIX platform for its Web Service and Application B uses Apache Axis on UNIX platform, is irrelevant. Both Web Services still communicate outside of their infrastructure using the same protocols.

What is Opera Web Suite?

Opera Web Suite (OWS) use Web Services technologies to support data transfer between client travel applications and the Opera Reservation System[®]. OWS is a collection of Web Services that provide access to key functionality on the CRS. The Web Services available through OWS provide access to a wide variety of travel-related functionality. As OWS evolves, additional CRS business functions will be abstracted and encapsulated into separate Web Services. OWS currently includes the following services:

Web Service	Description
Information Service	<p>Provides the functionality of the listing values as a Web Service. Business functions include transactions for:</p> <ul style="list-style-type: none"> • Query for List of values – Retrieves the list of values, specified by keyword related to common data such as Airlines, Hotel/Chain Codes, Guarantee/Credit Card types, Country/City information, Name/Address/Phone types and any other useful information regarding to all other services input to be satisfied. • Query for Hotel information – Retrieves the contact and detailed information about a hotel by specified hotel code. • Query for Rate information – Retrieves the detailed Rate Plan information such as Description, Commission and additional details about a specific hotel and rate code combination. • Currency Converter – Useful method to convert from one currency to another.

Web Service	Description
<p>Availability Service</p>	<p>Combines hotel, room stays and rate availability functions into one call to the CRS. Currently, this service supports</p> <ul style="list-style-type: none"> • Regional Availability by search code – Search the Opera database properties by geographical region, CRO, Chain code or country. The information in the response includes property codes, property status (open or close) and a real time rate range! • General Availability for a single property – Search the Opera database for the rate rooms available for a specific date range, number of persons for all public rates or include a specific rate code or access code to retrieve non-public rates. The response will include available room/rates, pricing information and indicators of rate changes (if applicable) amongst other data • General Availability by Promotion Code for a Single Property – Search the Opera database submitting a promotion code in a General Availability request to retrieve the availability of that promotion code within a date range and rest of mandatory items for a General Availability request message • General Availability by group code (Allotment) – Search the Opera database submitting a group code in a General Availability request to retrieve the availability of that group code within a date range and rest of mandatory items for a General Availability request message • Detail Availability for a Single property, rate and room code – Search the Opera database for the rules that are applicable to a specific room rate combination. The response will include, tax and commission information, rate change specific information if applicable, property information and marketing information. • Add-On Package Availability – Search for available packages that are applicable to a reservation. The response will include a list of package elements including rate, currency and description of the packages. • Inventory Availability – Search for available inventory such as rate restrictions and room occupancies that are applicable to a hotel. The response will include a list of calendar days including restrictions and inventory elements.
<p>Reservation Service</p>	<p>Combines booking functions into a call to the CRS. Currently, this service supports</p> <ul style="list-style-type: none"> • Create booking – Insert a booking in Opera with necessary data to complete a booking that includes, guest name, credit card information, property code, arrival and departure dates and number of persons in party. A successful response will include a confirmation number, details of the booking such as the cancellation policy, the guest name is echoed back and rate changes if applicable • Retrieve booking – As the name suggests, this is the capability to retrieve a booking with all of its associated information. • Update booking – This message allows users to update confirmed bookings in the Opera database. Update items such as arrival and departure dates, number of persons in party,

Web Service	Description
	<p>room type (inventory and pricing will be validated) and comments. A successful response will confirm the changes and echo back the new reservation elements</p> <ul style="list-style-type: none"> • Cancel booking – As the name suggests, with this message users can attempt to cancel a confirmed booking in the Opera database. Please note that cancellations can be subject to penalties. A successful response will contain a cancellation number • Future booking summary – The future booking summary request allows a guest to retrieve a list of his or her reservations that have arrival dates in the future. Alternatively, a future booking summary list can also be requested for a company using a registered Corporate ID. A successful response includes details such as confirmation numbers, room and rates that are booked and rate amounts amongst other information. • Email Confirmation – Sends an email with the contents of the reservation by specified Confirmation number. • Retrieve booking packages – Retrieves all add-on packages attached to a reservation by specified confirmation or leg number. • Daily point updates – Updates the daily points for an existing booking for point updates. • Retrieve a booking for point updates – This method will retrieve bookings available for membership awards by specified property details for which points can be earned. • Add/Update packages – Add or update packages to an existing reservation based on confirmation number, leg number and product code. • Delete packages – Remove add-on packages from an existing reservation based on confirmation number, leg number and product code.
<p>Name Service</p>	<p>Provides the Profile management including</p> <ul style="list-style-type: none"> • Create/Fetch/Update profile – Register profile in the Opera database. This message allows users to enter items such as name, address, phone and passport. A successful response will return a unique name identification number for the profile. • Add/update/delete/Fetch address to a profile – There are four distinct APIs to do the Add, Update, Delete and fetch of Addresses of a profile. The Add requires the unique profile name identification plus the data elements that are being inserted such as street number, city and country; a successful response of an insert address is a unique address ID. The Update allows changing elements such as street number, city, state, and country; the update requires the unique address identification number plus the data elements that are being updated such as street number, city and country. A successful update will return a success flag. The Delete (deactivate) address API only requires the address identification number. The fetch requires the unique profile identification number. A successful response will return a list of addresses attached to the profile. • Add/update/delete/Fetch Phone number(s) to a profile –

Web Service	Description
	<p>There are four distinct APIs to Add, Update, Delete and fetch phone numbers from a profile. The Add requires the unique profile identification number plus the data elements that are being inserted such as the phone number, phone role, phone type and primary flag indicator; a successful response includes a unique phone identification number. The Update requires the unique phone identification number and the elements that are being updated. The delete (deactivate) phone requires the unique phone identification number. The fetch requires the unique profile identification number. A successful response will return a list of phone numbers attached to the profile.</p> <ul style="list-style-type: none"> • Add/update/delete/Fetch email to a profile – There are four distinct APIs to Add, Update, Delete and fetch emails to a profile. The Add requires the profile unique identification number and the data elements that are being added such as email address and primary flag indicator; a successful response will return an email address unique identification number. The update requires the unique email address identification number and the data elements such as email address, primary flag and display sequence. The delete requires the unique email identification number. The fetch requires the unique profile identification number. A successful response will return a list of email addressees attached to a profile. • Add/update/delete/fetch Credit card(s) to a profile – There are four distinct APIs to Add, Update, Delete and Fetch credit cards to a profile. The Add requires the profile unique identification number and the data elements that are being added such as credit card type, credit card number, expiration date, credit card name and sequence. A successful response will return a unique credit card identification number. The update requires the unique credit card identification number plus the data elements to be updated such as credit card type, credit card number, expiration date, credit card name and sequence. The delete requires the unique credit card identification number. The fetch requires the profile unique identification number. A response will return a list of credit cards attached to the profile. • Add-update/delete/fetch Passport information to a profile – There are three distinct APIs to Add, Update, Delete and Fetch Passport information to a profile. The Add requires the profile unique identification number plus the data elements that are being inserted such as Passport number, issue date and place of issuance. The update requires the profile unique identification number plus the data elements that are being inserted such as Passport number, issue date and place of issuance. The Add and Update Passport APIs are exactly the same. The Delete Passport requires the profile unique identification number. • Add/update/delete/fetch Guest card information to a profile – There are four distinct APIs to Add, Update, Delete and Fetch Guest card information to a profile. The Add requires the profile unique identification number plus data elements being inserted such as membership type, membership card, membership level, and name on card and expiration date. A successful response will return a unique card identification number. The update requires the Card unique identification number plus the data elements that are being updated such as

Web Service	Description
	<p>membership type, membership card, membership level, and name on card and expiration date. The Delete requires the card unique identification number. The fetch requires the profile unique identification number. A successful response will return a list of cards attached to the profile.</p> <ul style="list-style-type: none"> • Add/delete/fetch preferences to a profile – There are three distinct APIs to Add, Delete and Fetch preferences from a profile. The Add requires the profile unique identification number plus data elements being inserted such as preference type and preference value. The Delete (deactivate) requires the unique profile identification number, preference value and preference type. The Fetch requires the unique profile identification number. A successful response will return a list of preferences attached to the profile. • Add/update/delete/fetch comments to a profile – There are four distinct APIs to Add, Delete, Update and Fetch comments from a profile. The Add requires the profile unique identification number plus data elements being inserted such as the note title and the note (or comment). A successful response will return a unique comment identification number. The update requires the profile unique identification number and the comment identification number plus the data elements that are being updated such as the note title and the notes (comments). The Delete (deactivate) requires the unique profile identification number and the comment identification number. The Fetch requires the unique profile identification number. A successful response will contain a list of comments that are viewable by the guest • Add/update/delete/fetch privacy to a profile - There are three distinct functions to add or update, fetch or delete privacy settings from a profile. The insert and delete requires the profile unique identification number plus data elements being modified such as the privacy options – Promotions, Market Research, Third Parties, Loyalty Program and the type. A successful response will return a success indicator.
<p>Security Service</p>	<p>Provides the access to all other web services and includes Authentication related functions. Currently supports:</p> <ul style="list-style-type: none"> • Authentication of Membership number/PIN – This message requires the membership number, last name and password (PIN). A successful response will return the unique profile identification number. • Authenticate non-registered user – This message allows non-registered users to obtain their unique profile identification number by submitting their last name and booking confirmation number. A successful response will return the unique profile identification number. • Update PIN and password – This API allows registered users to submit their membership number, last name, old password and new password and modify it in the system. • Update secret question and answer – This API allows registered users to update their secret question and answer by submitting their unique profile identification number, a secret question code (from Fetch Question List) and the secret question answer (free form). • Validate secret question – This API allows registered users to

Web Service	Description
	<p>retrieve their password/PIN by submitting the unique membership number, their last name and the secret question answer. The successful result will return the password/PIN.</p> <ul style="list-style-type: none"> • Fetch Question List – Retrieves a predefined list of secret questions available. • Create User – Creates a new user in the system by provided user name, password and security identification number. • Add/Update/Fetch/Delete Application User Profiles - There are four distinct APIs to do the Add, Update, Fetch and Delete of user profiles. The Add requires the user name, password and expiry date plus the data elements that are being inserted such as customer type, addresses, preferences, phones, emails, comments, user group; a successful response of an Add user profile is a unique name identification number. The Update allows changing the profile elements; the update requires the user name, password and expiry date plus the data elements that are being updated. A successful update will return a success flag. The Fetch allows retrieving the profile details of existing application user. The Delete (deactivate) user profile API only requires username and password. A successful response will return a success flag. • Login Application User with existing profile – Authorizes existing profile to use the system and its functions in a way the profile has rights to do so. No input parameters are required for this operation as all of the information is contained in the header section of the request. A successful response will return a success flag; user license key and security id (customer or company id) that can be used in future calls to all the existing services. Internet implementers of OWS are required to use this function and pass the parameters further to other services in order to process a successful operation.
<p>Membership Service</p>	<p>Allow clients to access data on hotel loyalty programs and supports the following functions:</p> <ul style="list-style-type: none"> • Membership transaction summary – This API allows members to submit their unique profile identification number and optional begin and end dates. A successful response will return a transaction list that contains data elements such as transaction ID, statement ID, posting date, transaction type, and transaction type description. • Fetch list of statement references – Retrieves a list of references associated with a membership card. The input can be an Opera internal ID or an object representing a membership card (containing card type card number, etc.). The response is a list of statement reference numbers and dates, which can be used to fetch membership statement. • Fetch Membership Statement – This allows a customer to retrieve a hotel loyalty program statement. The input parameter is a reference statement ID, a statement date and the internal ID for the membership card obtained by the list of references method. The return message contains a Statement that lists all the transactions covered under the said statement as well as the related statement details (cut-off date, beginning balance, ending balance, etc.) • Add/Fetch/Delete promotional subscriptions – There are three distinct APIs to insert, delete and retrieve promotional

Web Service	Description
	<p>subscriptions attached to a membership profile. The request requires an Opera internal membership ID and the response acknowledge a confirmation result for Add or Delete methods and a list of promotional subscriptions details for Fetch.</p> <ul style="list-style-type: none"> • Fetch Product/Rate/Room Upgrades Awards – There are three APIs to retrieve product, rate or room category upgrade awards based on affiliation information about the awards such as membership type, level and date range, this method displays a list of awards, their details and the points that can be earned. • Fetch Next Card Number – Retrieve the next membership card number. This API allows custom implementations for a list of membership cards. • Consume Points – Accumulates customer’s points for certain rate or product for a loyalty program based on membership ID and award details. • Consume Points Others – Accumulates customer’s points for other hotel defined programs based on membership ID and award details.
Stay History Service	<p>Provides the history information about guest stay in the hotel.</p> <ul style="list-style-type: none"> • Guest can retrieve stay history – This message requires the unique profile identification number. A successful response will contain a list of past reservations that have a status of Checked Out or that have been cancelled.

For more specific details of all the input and output parameters of all the described web services above, please refer to **Learn More About Opera Web Suite** section of this document.

What are the benefits of Opera Web Suite?

By standardizing communications between our customers, Central Reservation Systems, Opera Reservation Systems and Opera Web Suite offer a number of potential benefits for designing, implementing, and marketing travel-related applications. By combining existing Opera products with new technology standards, OWS offers a more streamlined and robust method for designing travel applications.

The benefits of OWS include:

- Platform independence.
- Encapsulated business logic with the CRS.
- More connectivity options.
- CRS communications components hosted by Micros-Fidelio.
- Flexibility and Extensibility
- International Scope
- Enhanced Availability Features
- Flexible profile management
- Stay History services
- Membership services
- Advanced logging

Platform independence

Because OWS adheres to cross-platform Web Services standards, OWS does not restrict the environment for developing or deploying client applications. OWS client applications can be deployed in any environment that supports HTTP (Hypertext Transfer Protocol), including Windows® and UNIX, without requiring bridges or specialized interfaces. In addition, a wide variety of languages and development developer toolkits can be used to design and deploy client applications for OWS.

Opera business logic encapsulated with the CRS

As OWS evolves, more and more Web Services will encapsulate the business logic needed to communicate with the CRS into a more streamlined, intuitive framework. As a result, developers focus on writing their applications rather than communicating with the CRS. The streamlined usability of OWS could potentially reduce development time by 30 to 70%, depending on the type of application and development environment. The web services call Opera logic, this means that an inventory query done via the hotel call center will return the same availability to the web user. There is only one database to setup with common set of booking rules applicable to both web and call center-originated reservations. There is only one database to set restrictions, to set inventory counts and rate periods with associated amounts that will apply across the board to different sources of availability and booking messages.

More connectivity options

Client applications can connect with OWS using either the Internet or a dedicated connection. Customers can select the type of connectivity that is most appropriate for their requirements and resources.

CRS communications components hosted by Micros-Fidelio

Previously, developers needed to include complicated code in their applications for identifying and routing transactions with the CRS. OWS streamlines communications with the CRS for customer-hosted applications and hosts any required components at Micros-Fidelio data centre, which internally supports identifying, routing, and transforming transactions with the Micros Fidelio ORS system (CRS).

Flexibility and Extensibility

OWS specifications provide travel service providers with the flexibility they need to develop, test and deploy new services, the minimum necessary functionality to provide reliable interactions between customers and the systems owned and maintained by the companies serving them.

As the versions of our ORS and Web Services evolve there are plans in order to ensure that incompatibility issues are kept to a minimum.

International Scope

OWS was designed to be Multi-language platform, supports the Unicode character set standards, include ISO countries, ISO currencies and city codes, based on United Nations code for trade and transport locations.

Enhanced Availability Features

In addition to traditional availability for a single property, OWS offers back-end functionality within Opera to retrieve real time availability of hotels grouped by geographical regions, cities, CRO and Chain codes. The real time regional availability response includes the available status of a property, a minimum and maximum rate range and finally, it also includes property information. Traditional availability requests for a single property include items such as arrival date, number of nights, number in party and possibly a specific rate code. OWS expose Opera availability search capability to a promotion code or a group code, introducing the concept of allotment sales to Micros Systems electronic distribution functionality!

Flexible profile management

Allow users to register in your database! Give them option to enter their preferences, credit cards and contact information. All of the Opera functionality behind the OWS can be used independently, allowing hotels to control what information they would like to capture from their registered profiles!

Stay History services

This feature allows guests to retrieve past booking information with the condition that they must be checked out or cancelled. This in itself is a great offering from hotels to their frequent travellers: to be able to look at the history of their business on the property or properties where they have stayed.

Membership services

OWS exposes the capability that Opera has to list a member's summary of statements, it also allows members to drill into each of the statement for specifics and further, it allows members to retrieve transactions that are not yet part of any statement, which means that they can have an up to date status of the points they have earned

Advanced logging

OWS incorporates advanced logic capabilities that allow users to have distinct count of message types and processing times.

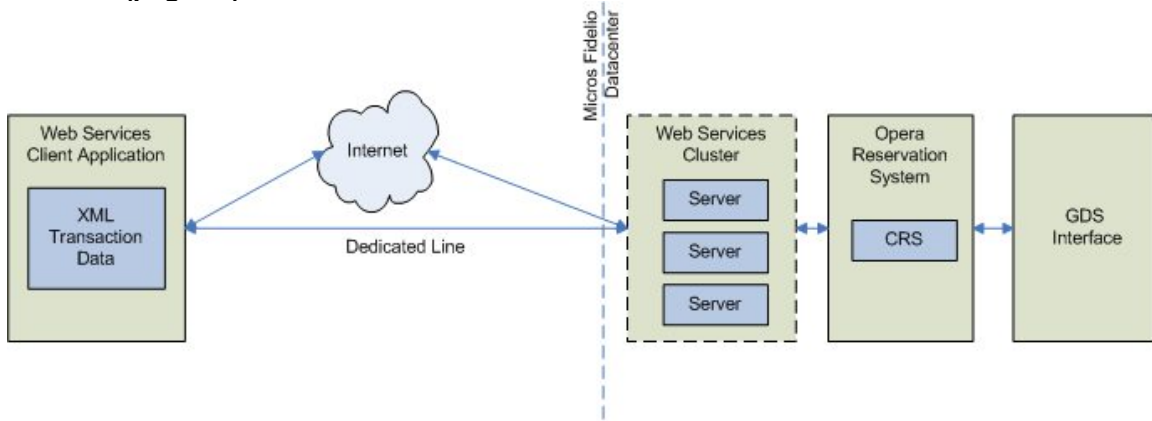
How Does Opera Web Suite Work?

Opera Web Suite combines GDS and CRS interface from ORS with Web Services technologies. The OWS product performs two key functions:

- Managing communications between client applications and the ORS.
- Converting between XML data from client applications and ORS proprietary structured data formats, which are used by the CRS and/or GDS depending on the channel used.

However, with OWS, the components are hosted by Micros-Fidelio on secured servers. By hosting their OWS interface at Micros-Fidelio, customers do not need to install, configure, and maintain components on-site.

The following graphic shows the basic flow of information in a transaction that uses OWS to communicate with the CRS and travel applications. For more information about the details of data transfer in a Web Services environment, please refer to **Appendix A: Know-How on Web Services (page 19)**.



The following description explains how a message is sent from a client application to the CRS.

Sending Requests

The client application creates a message with the request data in an XML (Extensible Markup Language) format.

Client applications can be written in any language or platform that supports Web Services technologies or protocols. See *Requirements and Recommendations* (page 16) for further details about supported environments.

Regardless of the programming language or platform, the application code must use XML for the actual request data that is sent to the CRS. This XML transaction is wrapped in a *SOAP envelope with a specific header and body*. SOAP (Simple Object Access Protocol) is XML data that directs the message to be delivered to the correct Web Service, and allows the message to be appropriately processed by the Web Service.

Network Connection: Public Internet or Private Dedicated Connection

The message is sent from the client application to the OWS servers, which are hosted by Micros-Fidelio.

Messages in Web Services are sent via HTTP (Hypertext Transfer Protocol), which is a common method of transferring HTML and XML data. OWS can be supported through HTTP used on either a public Internet connection or a private dedicated connection.

The amount of network bandwidth is the most important factor in planning for adequate capacity. Regardless of the type of network connection, the size of the connection has the most direct affect on the speed and capacity of the client application.

Web Services Servers

The Web Services servers receive and process the request message.

The SOAP message that was attached to the request message channels the request to appropriate Web Service, and essentially instructs the Web Service in how to process the message. A special header is included with the XML application data to identify the sender of the transaction to the Web Service.

Some Web Services messages end at the CRS servers, while others are forwarded to the GDS servers to be converted and processed by the CRS. How messages are handled depends on the type of origin and destination. For example, origin specified as GDS and destination specified as WEB, which will then present GDS availability on the WEB channel. The decode request and response is handled completely within the service. However, a GDS availability request requires additional processing by the CRS and the GDS. In this case, the Web Services act as a sort of intermediary for processing the availability request and response data.

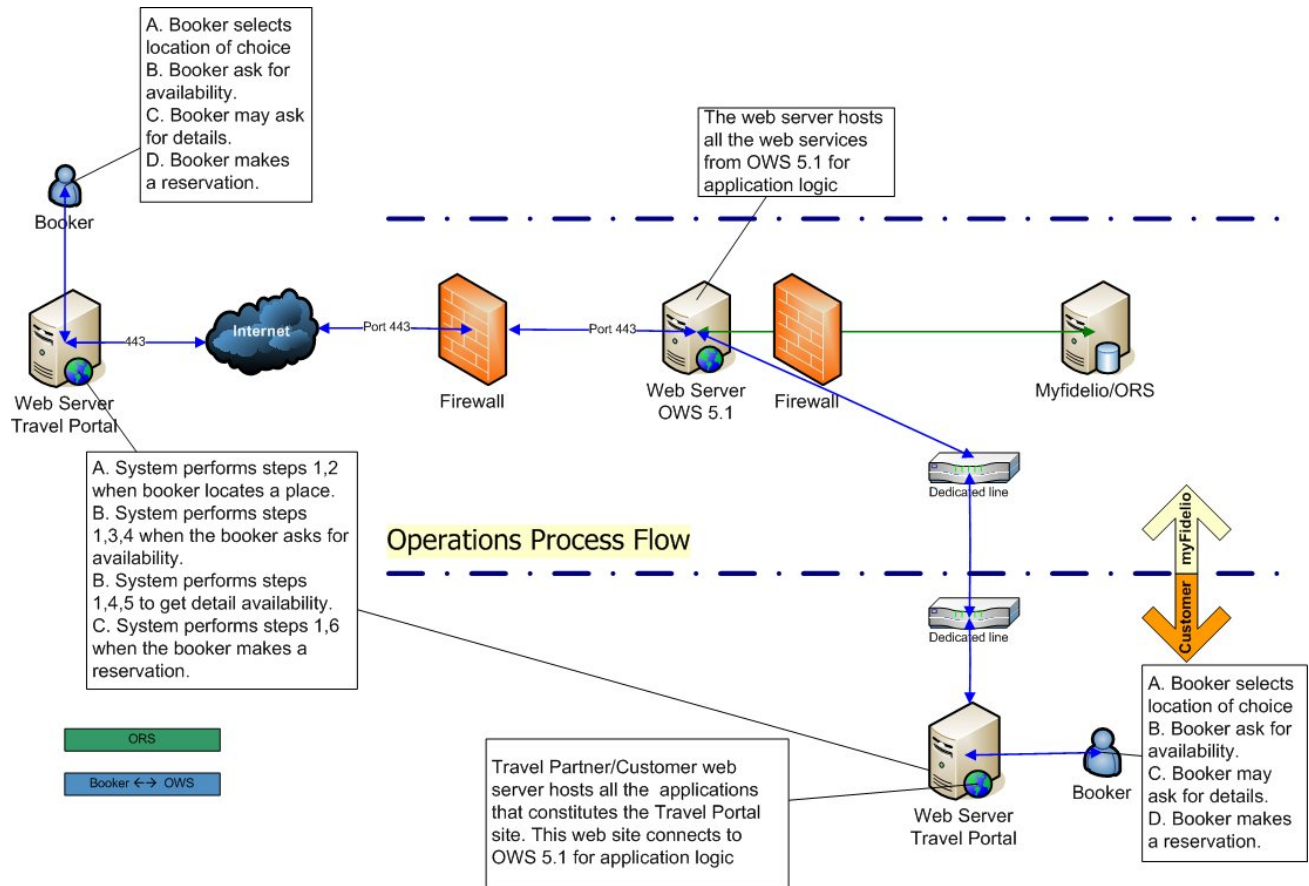
Retrieving Responses

The response message follows the same path as the request message, in reverse. The proprietary structured data response is sent from the CRS/GDS to OWS, which wraps the response in a SOAP envelope and forwards the message back to the client application. Again, the response is sent via HTTP through either the Internet or a dedicated connection. After the client application receives the message, it takes XML response data and displays that data within the client application.

The Booking Process

A complete Booking Process from client application to OWS is based on request-response schema explained above and follows the steps performed by the Travel Partner/Customer's system requested by a booker:

1. Client application authenticates itself using the Security service.
2. Selects City/Region/Chain by making a call to Information service.
3. Ask for a list of available properties by using Availability service.
4. Selects a specific Property and sends a General Availability Request using the Availability service.
5. Selects chosen Rate Room for Property and sends Detail Availability Request using the same Availability service.
6. Calls Create Booking using the Reservation Service and retrieves the response indicating the booking details.



Requirements and Recommendations

Planning for OWS includes familiarity with not only the required technologies and skill sets, but also preliminary planning to design a client application that can appropriately support Web Services for the expected amount of traffic within the scope of available resources.

OWS Standards and Protocols

OWS conforms to the following industry standards and protocols.

Message Envelope	SOAP 1.1	http://www.w3.org/TR/SOAP/
Message Transfer	HTTP 1.1	http://www.w3.org/Protocols/
Encryption	SSL 3.0	http://home.netscape.com/eng/ssl3/index.html
Web Service Description Language	WSDL 1.1	http://www.w3.org/TR/wsd/
Data Transfer	XML 1.0	http://www.w3.org/XML

Tools and Platforms

Any development tools or platforms that are compatible with the above standards and protocols can be used. These tools and platforms include, but are not limited to:

- Microsoft.NET
- IBM WebSphere® Studio Application Developer
- Apache Axis

Developer Skills

Developers should have familiarity with XML Web Services, as well as the standards and protocols listed in the in this section. Particularly when using the XML Web Service, knowledge of the CRS/OTA business model is also helpful.

Connectivity

OWS can be supported through either an Internet connection or a dedicated connection. Internet connections must be secured via an SSL (Secured Socket Layer) protocol, as well as the security functions provided for OWS. Because Web Services formats and standards are not limited to use on the Internet, the same protocols apply to the message, regardless of the type of network connection.

Security

The OWS servers are secured, and access is granted only to the specific Web Services for which the client is licensed. Internet connections to OWS must also be encrypted via an SSL (Secured Socket Layer) protocol. For dedicated connections, security methods for the connection must be defined in the client application or by using Point-To-Point tunnelling protocols with appropriate encryption methods. Because Web Services formats and standards are not limited to use on the Internet, the same protocols apply to the message, regardless of the type of network connection.

Capacity Planning

All capacity planning decisions should initiate from the projected scope and requirements of the client application. Capacity relates to the functionality of two basic components:

- The design and required functionality for the client application.
- The size and type of network connection.

The expected volume of Web Service calls is the key factor in capacity planning. Issues for determining expected volume include:

- The expected look-to-book ratio between requests to the CRS and actual bookings.
- The expected number of transactions.
- The types of Web Service calls planned, and their expected message sizes.

The most appropriate type of connection depends on the requirements of the client application and the available resources. Dedicated connections are generally faster and have more reliable availability, while Internet connections are typically less expensive. However, many of these factors vary depending on the environment. For example, in many locations dedicated connections are prohibitively expensive for some organizations, while in other locations Internet connections may actually be more reliable than the local telecommunications required for a dedicated connection.

Support for Opera Web Suite

Technical support is provided during the development of new applications and modification to existing applications. Micros-Fidelio supports OWS only; the programming technique, language, or environment is not supported.

Online Help

OWS include a Software Developers Kit that provides theoretical and practical information for using Opera Web Suite, including:

- A general overview of Web Services.
- Descriptions of each OWS Service.
- Recommendations and requirements for application design and security.
- Getting started guides and sample transactions.
- Detailed information about how to use each web service, including the individual transactions.

OWS Sample Site

Opera Web Suite will offer a Sample Web Site very soon.

The Sample Site will allow you to:

- Test Web Service requests for hotel availability and reservations.
- Read Overview of OWS architecture.
- View and download sample code (SDK) for several languages and platforms.

The features available on the Sample Site depend on your level of access:

- Everyone can use the sample requests and read the *Overview*.
- Prospective customers, who receive a preliminary user name and password from Micros-Fidelio, can also review the *Documentation* section, including the OWS Help System.
- Licensed customers, who receive a user name and password, can also access the sample code in the *View Code* section.

Learn More About Opera Web Suite

There are several ways to learn more about Opera Web Suite.

Web Site

Additional information about Opera Web Suite is available at <http://webservices.micos.com/ows/5.1/documentation>

Contact Us

Prospective customers can contact one of our Micros-Fidelio regional offices.

Customers in Europe, Africa & the Middle East should contact:-

Micros-Fidelio Software Deutschland GmbH & Co.KG

Regional Operations center
Euro Center
Europadamm 2 –6
Neuss, Germany
Phone: +(49) (2131) 137-0
Fax: +(49) (2131) 137-702

Customers in Asia Pacific Region should contact:-

Micros Fidelio Software

Regional Operations Center
Sydney
Australia
Phone +61 2 9485 1000
Fax +61 2 9485 1099

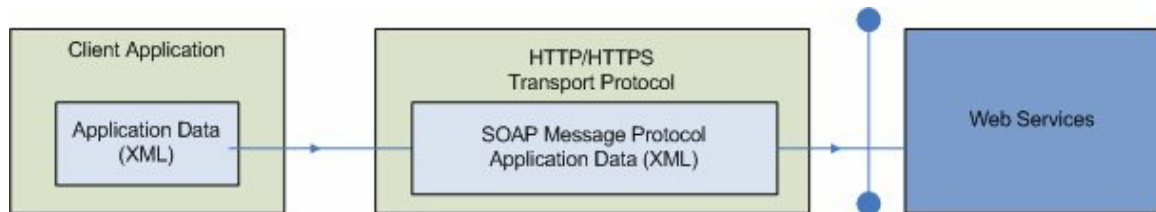
Customers in the US should contact:-

Micros Systems Inc

7031 Columbia Gateway Drive
Columbia, MD 21046-2289
Phone +1 443 285 6000
contact@micros.com

Appendix A: Know-How on Web Services

Understanding the underlying concepts behind Web Services helps to better illustrate why Web Services streamline and transform communication with the CRS. For all of the terms that surround Web Services, the basic premise of Web Services is not very complex. The following diagram shows the basic flow of data between a client application and a Web Service.



XML (Extensible Mark-up Language)

XML is the format for message data used with Web Services.

What is XML?

Like its “cousin” HTML, XML is not a programming language, but rather a system for tagging chunks of data so that they can be sent and received in a defined, self-descriptive fashion. HTML tells you how the data should *display*. XML not only controls the display of data, but also tells you *what* the data is and *how* to use the data.

XML is commonly used across the Internet, but can also be used to transport information in a number of other environments.

XML has several particularly useful qualities that can be applied to Web Services:

- *Customizable tags* that can be used to precisely define content for a specific application. The tags, together with the data described by the tags, form an XML *element*.
- Developers typically use an XML schema to describe specific XML data and define how to use that data.
- *Data storage* within the application itself, which can increase product usability, reduces transaction time, and decrease server usage.
- *Separation of display and content*, in which changing the display of content does not affect the content itself.

How do Web Services use XML?

XML is used in two main ways for Web Services:

- The actual application data that is sent between systems is formatted in XML. For OWS, the data that is sent between a client travel application and the Opera Web Suite is in XML.
- Message protocols and supporting data that are used to send the application data are also written in XML.

SOAP (Simple Object Access Protocol)

SOAP is a specific kind of XML message that is “wrapped around” the XML application data. SOAP directs the application data to the Web Services, and defines how the Web Service processes the data.

Written in XML, the SOAP message is then wrapped in a *SOAP envelope* around the application data, also written in XML. SOAP indicates:

- How a message is being sent
- Where the message is being sent
- What kind of message is being sent

For example, in OWS, a SOAP message for a hotel general availability request would indicate that the XML data for the transaction should be sent:

- From the client application via HTTP (the standard for Web Services data transaction).
- Specifically to the OWS Availability Service (which processes the requests).

WSDL (Web Services Definition Language)

The WSDL provides a compilation of the functions that can be performed by a specific Web Service.

The WSDL is another XML document that contains the XML schema about how to access and use the functions for a specific Web Service. For example, the WSDL for the Availability Web Service contains information about functions such as Regional Availability transactions, receiving transactions, and beginning sessions. When a SOAP message is sent to the Web Service from the client application, the data about how to use that Service has been obtained from the WSDL. Typically, the WSDL programmatically interfaces with the software developer toolkit. Before designing a client application, developers create a proxy (copy) of the WSDL, which then provides them with the Web Service parameters need to design their application and data structures.

HTTP (Hypertext Transfer Protocol)

HTTP is the method by which XML application data and SOAP data are sent between client applications and Web Services.

HTTP is a communications protocol for transferring data that is already commonly used on the Internet. HTTP is used extensively on the World Wide Web to transfer HTML from Web servers to Internet browsers. The **http://** prefix in front of a web site address indicates that HTTP is the protocol being used by that web site.

Because XML and HTML are related, XML can also use HTTP as a transfer protocol. Web Services do not strictly require the use of HTTP; however, it is currently the most commonly used transfer protocol for Web Services.

While HTTP is used most visibly on the World Wide Web, its use is also not limited to the Internet. For example, OWS clients also have the option of using HTTP across a dedicated connection or via the Internet.