Winoffice Prime Web Service Architecture overview

Inhalt

Revision History

Version	Date	Modified By	Summary of changes
1	23.03.10	MMI	Draft
2	06.10.10	MMI	Updated for the Web Service API
3	02.02.17	MMI	Removed description of the Prime server proprietary protocol; Updated description of the POX security implementation

Definitions, acronyms, abbreviation

Definition	Explanation
WCF	Windows Communication Foundation
REST	Representation State Transfer
POX	Plain Old Xml
SOAP	Simple Object Access Protocol

References

[1] Service Stack http://code.google.com/p/servicestack

[2] WCF RIA Services http://www.silverlight.net/getstarted/riaservices/

Introduction

Winoffice Prime is an ERP system, which runs on Microsoft platform. In order to work with the system, users have to install a client application. The communication protocol between the server and the client is proprietary. In order to support mobile clients and enable easier integration with the third party systems, a layer of web services has been developed. This layer is at the moment used by the Winoffice iPhone client.

Overview of the Winoffice Prime architecture

Winoffice Prime is a client-server application (see Figure 1).

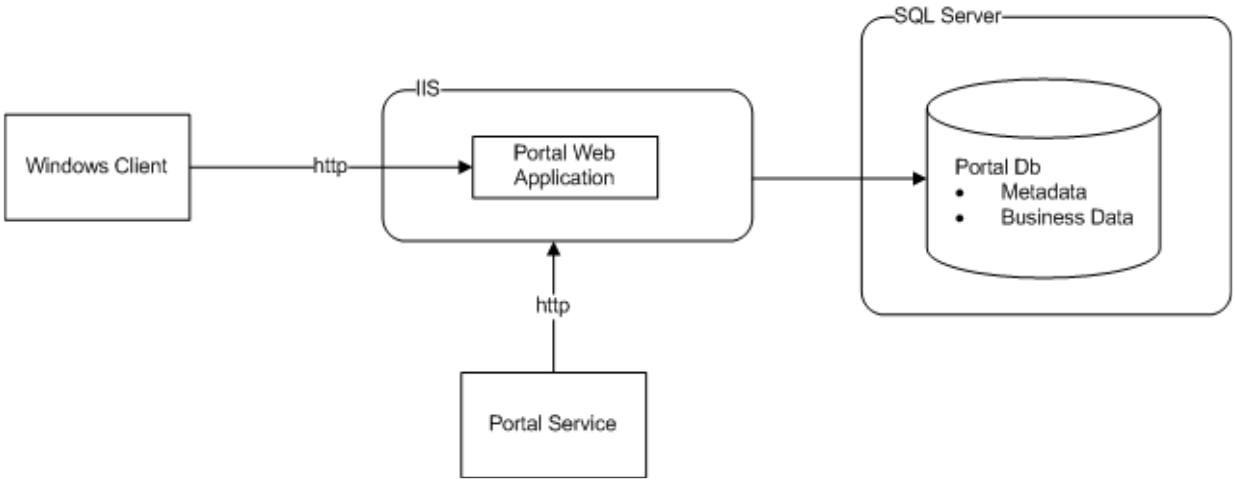

Figure 1 Winoffice Prime architecture summary

The Server part is responsible for retrieving and modifying data in the database, coordinating activities of different clients though logical locks and notifications, and executing scheduled activities, such as updating full text indexes, downloading and sending mails. The server layer is relatively thin, since it is responsible for low-level platform operations.

The Client part combines presentation and business logic layers. It is responsible for representing data to the user, accepting and validating user input as well as enforcing and executing business rules. The Client is started as a small startup executable, which downloads and keeps up to date assemblies and other resources that constitute the whole client application. Part of the business data (mostly dictionaries) is cached on the client as well.

Web Service Layer

It is not feasible to use proprietary Prime Client-Server communication protocol for integration scenarios, since it would require implementing significant part of the framework and business logic in the service consumer. An intermediate layer that can leverage capabilities of the existing framework and business logic has been implemented and exposed though a set of web services.

Web service layer expose only a subset of portal functionality. The services are grouped by business object type and usually consist of a set of CRUD operations and methods to list objects selected and ordered by certain criteria. The services are not automatically generated from business object definitions. They are manually programmed and along with the CRUD operations can offer a set of additional method. There is no one to one match between domain objects exposed over the services and portal business objects. Web service counterparts usually have significantly fewer properties.

Web services primarily target the development of light-weight clients and integration scenarios. They do not offer support for pessimistic locking, transactions or subscriber-publisher patterns, what makes them of limited use for writing complex business logic on the consumer side.

Implementation

The Web Service Layer implementation is based on WCF and allows flexible configuration. Two sets of endpoints are supported out of the box: SOAP with basic HTTP binding and POX with web binding. Deployment with the portal web application limits bindings to those, which are supported by IIS. It is however possible to use custom deployment in order to support other scenarios, for example SOAP over TCP or named pipes. SOAP end points are intended for integration scenarios. POX end points are intended for mobile clients.

Note: *In order to facilitate development of light-weight mobile clients, the current implementation does not provide a strict separation of data and there representation. For example, the concept of list items or reference properties targets visual representation of items.*

Note: *Web Api supports only simple scenarios for selecting and modifying a subset of Prime business objects. More complex scenarios where a consumer could define, which properties of an object should be selected, load object hierarchies with LINQ-like queries or update a set of objects in a single transaction are not supported.*

Read Operations

There is usually a method to load a list of items, selected and ordered by some criteria. The selection is limited by the number of items on a page and the first item to be included. The selection criteria are defined with a filter object:

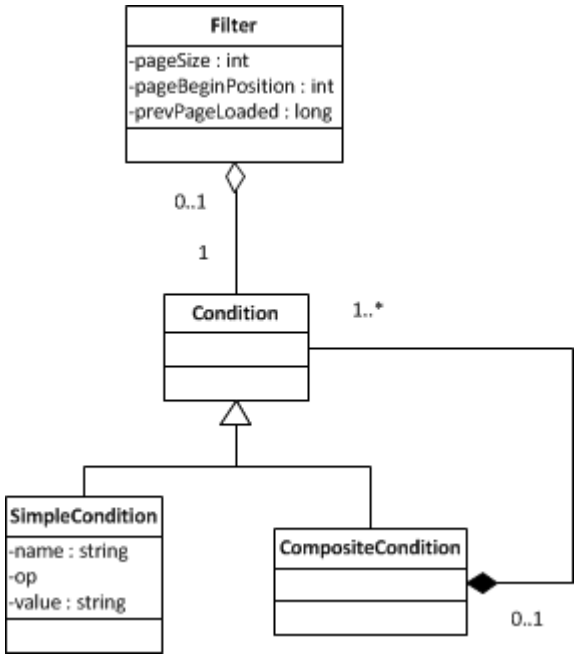

Figure 2 Filter and Condition classes.

Simple conditions are property name and value pairs linked with an operator to be applied to the value. For each object, there is a fixed subset of properties, which can be used for selection. Simple conditions can be combined with logical “And” and “Or” operators to form more complex criteria. Page and start item position are zero-based.

Selection methods return a collection of either generic list items or a specialized version of the list item. List items contain id, modification time stamp and short description of an item. Specialized version can contain additional attributes, like start date for an activity.

Additional convenience selection methods can be added to the service to allow consumer calling selection methods without constructing a filter object.

Get Details method allows retrieving complete objects by either id or a filtering criteria. Each object has at least two properties: id and modification timestamp.

Write operations

Web API usually offers three methods for objects to be modified: Create, Update and Delete. These methods return an object of type ModificationResult with the id and modification timestamp of the object. If the operation fails, a fault is returned instead.

Reference Properties

Along with simple properties, object can have references to other objects. For example, Sales Order references Address. In the database the reference is represented by the field “Account_ID”. Web Api replaces such fields with list items, which contain id as well as a short description of the item. This approach allows light-weight mobile client avoid querying web service multiple times in order to display an item with external references.

Complex Objects

Some business objects contain dependent objects, which cannot be modified independently and should be handled as a single complex object with its parent. For example, a sales order should be saved in a single transaction with its lines.

Lines are loaded via a call to GetSalesOrderLines method. The usual rules for filter apply. Create and Update operations accept a complete Sales Order with line collections. Each line has a diffgramStatus to let the server know, how the line should be saved.

Authentication

Web services impersonate a Winoffice Prime user and require authentication before any business operation is executed.

Different authentication mechanisms are used for SOAP and POX endpoints. SOAP authentication is based on Username-Password send in the SOAP envelop. POX endpoints require passing user credentials in a custom header "X-WinofficeAuthenticate" with each request, similar to basic authentication. Both endpoints rely on transport security (SSL) for the protection of user credentials against eavesdropping.

Note: *Basic authentication could have been used for both POX and SOAP endpoints. The problem is caused by the deployment of the service together with the portal web application in IIS. Built-in IIS HTTP authentication modules are hardwired to Windows accounts. It is possible to change this behavior (check http://custombasicauth.codeplex.com/), but would require significant deployment efforts and will not necessarily be allowed in hosting environments.*

Error Handling

Business errors are returned as faults serialized in xml:

```
<Fault xmlns="http://schemas.microsoft.com/ws/2005/05/envelope/none">
 <Code>
  <Value>server</Value>
 </Code>
 <Reason>
  <Text xml:lang="en-US"> The address id=[-2144341006] cannot be modified, since it has been
locked by another user.</Text>
 </Reason>
 </Detail i:nil="true">
</Fault>
```

Figure 3. Sample fault

Consumer should also be prepared to handle HTTP errors codes 4**, 5**

Client Development

For the development of windows platform, a library with entities and service contracts is distributed. At the time of this writing, the library is compiled for .NET 4.5.2

Proxy classes can also be generated from the service wsld, but in this case helper methods in objects will not be available.

The client can be also programmed directly against POX end point. Xml schemas for entities and service contracts are available in the form of wsdl. This approach is not feasible, if the development environment supports generation of proxy classes from wsdl.

Customization

WS-API can be customized by implementation of custom web services.

Sample Request-Response Sequence

The following sample show request and response exchange between the client and the server in order to retrieve address details:

```
POST /PrimeWebService/Address/GetAddressDetails HTTP/1.1
Content-Type: application/xml; charset=utf-8
Content-Length: 59
X-WinofficeAuthenticate: .....

<GetAddressDetails>
<id>-2144341006</id>
</GetAddressDetails>
```

Figure 4 . Sample request

```
HTTP/1.1 200 OK
Content-Length: 123
Content-Type: application/xml; charset=utf-8
Server: Microsoft-HTTPAPI/2.0
Date: Sun, 22 Nov 2009 07:43:42 GMT

<item>
<city i:nil="true"/>
<email/>
<fax/>
<id>2144341006</id>
<name>A&amp;amp;B</name>
<phone/>
<street/>
<web/>
<zip>5000</zip>
</item>
```

Figure 5. Sample response