Generic clients supported by Web Services

Xuesong Liu

Kongens Lyngby
10th April 2007

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Abstract

With an established Web Services Framework to expose the functionalities of Maconomy ERP system as Web Services (WS), Maconomy has an interest in the area of specialized client-server protocols, which could be leveraged to support generic clients in a Service-Oriented Architecture (SOA).

The main objective of the master project is to investigate two specific WS protocols for SOA - "Microsoft Information Bridge Framework (IBF)" and "Web Services for Remote Portlets (WSRP)", and evaluate the compatibility of Maconomy Web Services with them.

Finally, an IBF-based prototype was proposed and implemented successfully to demonstrate the feasibility of supporting a generic client with Maconomy Web Services.

Preface

This master thesis has been accomplished at the Informatics Mathematical Modelling (IMM), the Technical University of Denmark (DTU), from September 11 2006 to April 10 2007. The project constitutes the final work of the requirement for obtaining a Master degree in Computer Systems Engineering at DTU. The project was supervised by Associate Professor Jens Thyge Kristensen and assisted by Software Developer Nikolaj Oldager from Maconomy A/S.

First of all, I would like to express my deep-felt gratitude to my advisor, Jens Thyge Kristensen and Nikolaj Oldager for their advice, encouragement, enduring patience and constant support. In addition, a special thank goes to Bo Stig Hansen, the former manager of the Architecture department of Maconomy for offering the thesis topic.

I would also like to thank my dearest wife for encouraging and supporting me during my study. Finally, I would like to thank all my friends for always giving me advices and inspiring me in my life.

Lyngby, 10 April 2007

Xuesong Liu
s041382

Contents

List of Figures

Chapter 1

Introduction

With the continuing popularity of Web, Web technology is not only applied to the human-oriented interactions but also more and more applied to the software-oriented interconnection. Web Service (**WS**) is exactly a distributed computing technology adapted to the Web. Using XML-based standards and transport protocols to exchange data, Web Services enable users to connect different components even across domain in a platform- and language-independent manner. As an emerging technology that makes it possible for different systems to communicate and exchange data via the internet, Web services are more and more applied into the developments of enterprise applications.

By wrapping the existing enterprise application systems or components into Web services, the various data and functionalities of a preexisting enterprise application can be provided as a 'Service' which is always available with an exactly common interface to the different 'Consumers'. Accompanying with the wide adoption of Web services technology into the architecture of enterprise systems, the concept of Service Oriented Architecture (**SOA**) is referenced. "*A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activities. Some means of connecting services to each other is needed.*"[1]

In SOA, one of the typical interaction patterns is the Clients/Web Services inter-

action. Clients are envisioned to provide common user interfaces to functionality exposed by multiple Web services, possibly supplied by different vendors. Such clients are also characterized as "Composite Applications" or "Service Oriented Clients"(**SOC**).

The proposer of this thesis project, Maconomy has extensive experience in building generic clients. The clients are parameterized with specifications of data source, layout and interaction behaviour. The specifications are general enough that they can be (and have been) implemented across a wide range of GUI-technologies, including HTML, Java and Windows GUI-technologies.

To make such a generic client supported by Web services (the so-called SOC), it is a precondition that we can access the various data and functionalities of Maconomy business solutions by means of Web Services. Moreover, some specialized client-server protocols for Service Oriented Architectures are to be investigated so that an open standard could be leveraged to connect generic clients with Web services.

At present, Maconomy has developed a Web services framework which can expose any Maconomy system functionality as a Web service. Therefore, how the generic clients could be supported by Maconomy Web services just becomes the subject of this thesis project.

1.1 The thesis objective

The objective of this master project is to investigate and answer such a question: how to construct a generic client supported by Maconomy Web services with any existing WS standard?

To achieve this objective, two specific WS standards for building SOC are expected to be investigated. They are the "Microsoft Information Bridge Framework (**IBF**)" and the "Web Services for Remote Portlets (**WSRP**)". At the same time, the usability of Maconomy Web services in supporting SOC should also be evaluated.

Finally a prototype solution should be proposed and implemented to demonstrate the feasibility of supporting a generic client with Maconomy Web services.

1.2 Intended audience

In the view of the experience gained during the work on the thesis, it became apparent that the intended audience is to be software developers and project managers that are willing to start working on SOCs related projects.

In particular, the readers could be developers who are interested in building Microsoft Office-based enterprise solutions by using Information Bridge Framework.

Ideally, the experience of using Web services is recommended and understanding of most common Internet technologies is required. It is advisable that familiarization with XML technologies is done as a prerequisite.

1.3 Methodology

The study was performed by looking up Internet resources, publications and books as well as Maconomy internal documentations. The implementation of the sample solution was performed on two standard computers with Microsoft Windows Server 2003 installed.

Chapter 2

Background

2.1 Web Services

Web services infrastructure and standards are designed to make it possible that different software systems can communicate and exchange data via the Internet, especially without human interference. This kind of "machine to machine" interaction model is independent to any programming language, operating system and transport protocol.

Web Services interaction model mainly consists of three tiers which are respectively: Service Provider, Service Registry and Service Consumer. Figure 2.1 is shown to illustrate the model.

Firstly a Service Provider makes an implementation of a service and describes the service in a Web Service Description (WSD). Next, the service description is submitted to a Service Registry by that Service Provider and registered there as an entry along with other contact information. Once a Service Consumer interested in the usage of a particular service queries the Service Registry to obtain the WSD, the consumer can generate the appropriate message to communicate with that Service Provider over the standard protocol.

To support such a framework, Web services require the use of several related

XML-based technologies as shown in Figure 2.1. They are: Simple Object Access Protocol (**SOAP**), Web Service Description Language (**WSDL**), and Universal Discovery, Description, and Integration (**UDDI**). The following subsections are to explain them respectively.

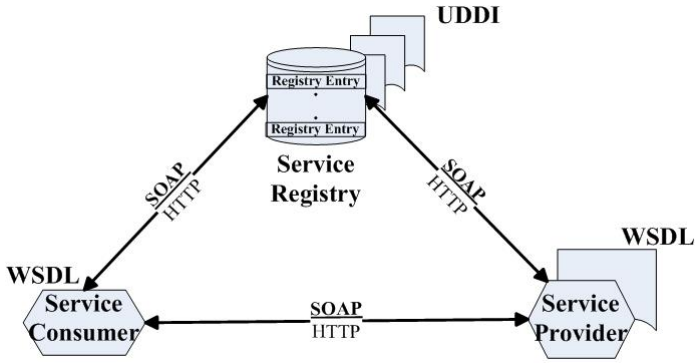


Figure 2.1: Web services model and related technologies

2.1.1 The foundation of Web Services - XML

Extensible Markup Language (XML) is the foundation on which Web services are built. XML messages provide the common document format by which different applications can talk to one another over a network. To operate a Web service, a user sends an XML message containing a request for the Web service to perform some operations. In response the Web service sends back another XML message containing the results of the operations. XML is also used to create the Web services technologies that exchange the data. In summary, "*XML provides the description, storage, and transmission format for data exchanged via Web services.*" [2]

2.1.2 WSDL

WSDL is described as "*an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.*" [3]

WSDL documents are the basis of interaction with Web Services. Their struc-

ture might be overwhelming at first but the syntax complexity is not immense. Each document is based on a XML schema which describes the XML instance. The most appealing characteristic of WSDL is its possibility for automatic generation of classes and methods in any given programming language.

The root element of a WSDL is identified by definitions. It may contain up to six different types of sub elements:

- **Types**: "*the data types - in the form of XML schemas or possibly some other mechanism - to be used in the messages.*" [2]
- **Message**: "*an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.*" [2]
- **Operation**: "*the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.*" [2]
- **PortType**: "*an abstract set of operations mapped to one or more end points, defining the collection of operations for a binding; the collection of operations, because it is abstract, can be mapped to multiple transports through various bindings.*" [2]
- **Binding**: "*the concrete protocol and data formats for the operations and messages defined for a particular port type.*" [2]
- **Port**: "*a combination of a binding and a network address, providing the target address of the service communication.*" [2]
- **Service**: "*a collection of related end points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.*" [2]

2.1.3 UDDI

UDDI is stated as "*a Web services registry and discovery mechanism, is used for storing and categorizing business information and for retrieving pointers to Web services interfaces.*" [2]

In other words, UDDI provides a means to publish the Web services on the Internet or Intranet to make the information of their existence to be available to service consumers.

On the other hand, it is also possible to make use of Web Services without publishing them in the UDDI registry by making a private agreement between a service consumer and a provider (sharing a specific WSDL file).

2.1.4 SOAP

SOAP (Simple Object Access Protocol) is

"*a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.*" [4]

The SOAP is based on XML messages. Figure 2.2 illustrates the three main parts of a SOAP message: the SOAP envelope, the SOAP body and the optional SOAP header.

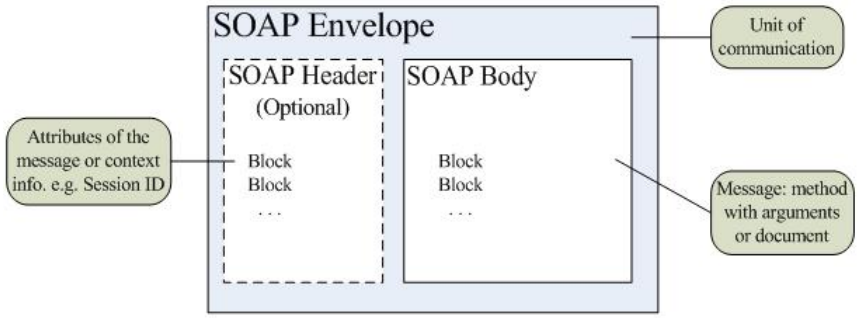


Figure 2.2: SOAP consists of three parts

- **SOAP Envelope**: The envelope marks the start and the end of the SOAP message. "*It expresses what is in a message, who should deal with it, and whether it is optional or mandatory.*" [4]
- **SOAP Header**: "*The header is optional, and can contain one or more header blocks carrying the attributes of the message or defining the qualities of service for the message. Headers are intended to carry contexts or any application-defined information associated with message, such as security tokens, transaction identifiers.*"[2]
- **SOAP Body**: "*The body contains one or more body blocks comprising the message itself.*" [2]

The use of SOAP is primarily related to object serialization. Serialization is a process of transforming a runtime object into a stream of data. Deserialization is the opposite process which recreates a runtime object from a stream.

How SOAP fits in the Web Services model is depicted in the Figure 2.3.

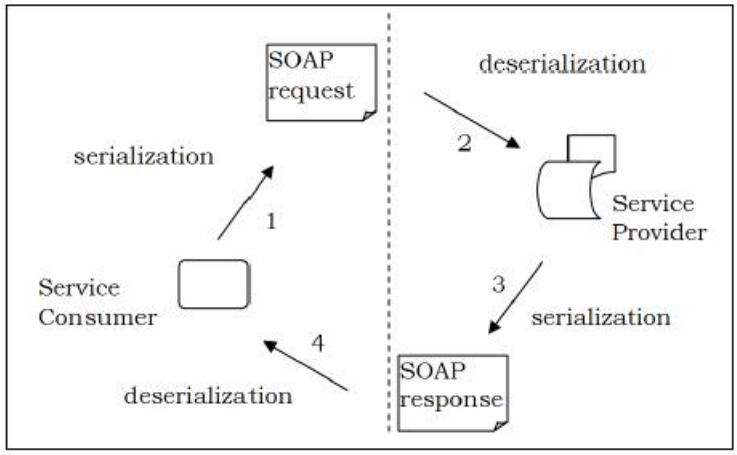


Figure 2.3: SOAP message flow

In summary, the SOAP takes the responsibility of delivering XML messages from one place to another over the Web. It is perhaps the most significant of all Web services technologies.

2.2 Service Oriented Clients

The Service-Oriented Client is just what the name implies - a client that is designed to interact with services. Using SOC architecture, developers can create rich Internet applications that make SOA data meaningful and useful to human beings. "*SOCs combine the broad deployment reach of the Web with the richness and intelligence of the desktop client software. For instance, SOCs work online or offline, connect with message-oriented middleware, and blend data, documents, forms, and audio and video streams-all in an interactive environment.*" [5] SOCs complement the capabilities of SOA and deliver rich, effective experiences to the user.

In the aspect of binding a client with service providers, SOCs are designed for federated data to come together, and then apart, easily. This means that SOCs

are envisioned to provide common user interfaces to functionalities exposed by multiple Web services, possibly supplied by different vendors so that the possible extension or substitution to services could be performed easily.

The theoretical model of the interaction between SOCs and Web services could be illustrated by Figure 2.4. In this figure, it is assumed that the WSDL file for a specific Web service has been obtained from a service registry or other location. The WSDL file is used to define both the request and response data transformations.

At the client side, an initial request in its native format is serialized into SOAP messages which could be delivered by SOAP handler to the intended Web service port as specified in the WSDL file. The opposite process is that the responded SOAP messages are deserialized into client-side runtime object which is just the response clients expect from Web services.

At the side of service providers, the corresponding Web service port receives the SOAP message and figures out which method to call. After that, the SOAP handler deserializes the SOAP message into server runtime objects that can be supplied as method arguments during invoking the implemented method. When that method returns, the output object is serialized to SOAP messages that are sent back to clients as the response.

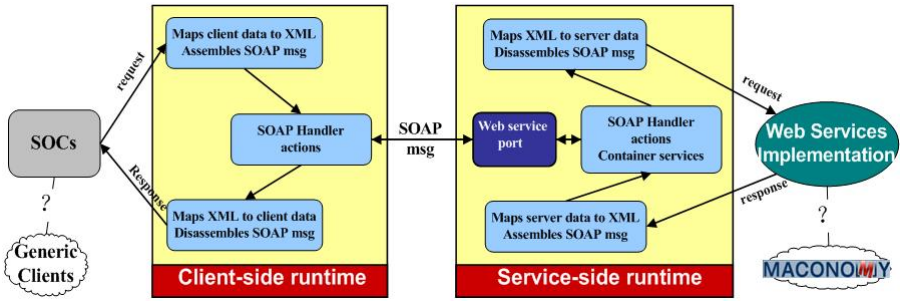


Figure 2.4: Interactions between SOCs and Web Services

In this project, the so-called generic clients are supposed to play the role of SOCs (as shown in the above figure) and meanwhile follow the client specifications that are also complied by the other kinds of Maconomy clients. To support the generic clients, Web services which can access Maconomy application data and functionalities are needed and will be discussed in the next section.

2.3 Maconomy Web Services Framework

To 'wrap' the existing Maconomy application functionalities into Web services, Maconomy has implemented a Web services framework to facilitate this transformation. Because Maconomy uses a domain-specific programming language - M-Script in the development of Maconomy Web services, it is better to first have an overview to Maconomy M-Script language as well as M-Script API.

2.3.1 Maconomy M-Script Language

The Maconomy M-Script language is based on data from a Maconomy system. An M-Script program, or simply an M-Script, is executed or rather interpreted by an interpreter.

The M-Script language is based on the syntax and semantics of the JavaScript language. It is in many ways similar to other Web programming languages such as PHP, ASP, PERL, and server side Java.

There are a number of good reasons for Maconomy to invent and use M-Script: M-Script is platform-independent and runs on Windows, IBM AIX, HP UNIX, Linux as well as Solaris. M-Script integrates seamlessly with Maconomy data. This means you get Maconomy's access control, access to all Maconomy's dialogs, and access to Analyzer reports etc., and all this as an integral part of the language. This could be difficult to obtain in the other public domain languages.

The detailed description about M-Script can be found in "M-Script Language Reference". [6]

2.3.2 What is M-Script Maconomy API?

The acronym API stands for "Application Programmer Interface", and the M-Script Maconomy API is an interface that allows the M-Script programmer to access data in the Maconomy database.

The API consists of a collection of functions and procedures that can be called from M-Script. All these subroutines reside in the M-Script module maconomy, which means that all subroutine names must be prefixed with "maconomy::".

A simple M-Script using the Maconomy API is:

```
#version 14
newsession();
maconomy::login("Administrator", "123456");
var echoReply = maconomy::echo("Hello Maconomy!");
print("Echo from Maconomy: ^1\n", echoReply);
maconomy::logout();
deletesession();
```

This script performs a login to Maconomy, calls the echo function of the API, prints the result, and finally logs out from Maconomy.

Dialog Interface

To avoid API calls corrupting the data in the Maconomy database, there must be a controlled way to read, update, insert, and delete data in the database. Dialog Interface was designed to give full access to Maconomy using the "dialog model". This means that data are read, updated, inserted, and deleted as if it was done through dialog windows in the Maconomy client.

Dialog Model imposes restrictions on the order of operations on a dialog. For example, it is not possible to perform a put operation without performing a new operation first. These restrictions are enforced by the Dialog State Machine.

Dialog State Machine is simply a device that keeps track of a state. From this state the state machine knows which operations are valid and which are not. Any open dialog is always in one of these four states: Initial, Exist, NewUpper and NewLower.

For instance, when a dialog is in the state of 'Exist', the operation - dialogPut-Lower can not be called from this state but only from the state of 'NewLower'.

The detailed description about Maconomy API can be found in "M-Script Maconomy API Reference". [7]

2.3.3 Maconomy Web Services

Basic components

The Maconomy Web Services Framework allows the M-Script programmer to expose any M-Script package as a Web service, including WSDL generation, SOAP handling, and HTTP GET handling.

The Web service framework is split into four different components:

- A few global packages for M-Script located in the namespace mscript::soap (installed by the standard M-Script).
- The framework files, located in <root>/MaconomyWS/Framework.
- The actual services located in <root>/MaconomyWS/<application-version>.
- The M-Script interpreter MaconomyWS.exe, which is just another copy of the standard M-Script executable.

Here <root> is the root of the Web service installation and <application-version> is an identifier for the application version.

Writing a Web Service

Writing a Web service can be as simple as writing a standard M-Script package and then placing it in a proper directory on the web server. The package itself then becomes the service and all of the public functions in it become operations of the service. It is almost as simple as that, the only extra thing to add is a public variable named **serviceDescription**. This variable must be an object that defines a few global properties of the service.

From a programmers view, this is all that is needed. The **WSDL**, **SOAP** and **HTTP GET** interfaces are generated automatically from the service package. Encoding and decoding of XML data is handled by the framework.

Assuming the Web service framework is installed on your web server with the base URL "http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe", you can now access your Web service description (WSDL) as "http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe/wsdl.ms?service=" followed with the name of a specific service. The output is an XML WSDL document ready to be used with either Microsoft's .NET tools or a similar Java tool set.

To expose an M-Script package as a Web service, you just need to place it in the <root>/MaconomyWS/<application-version>/Custom directory. Just move it into that directory, it is all that is needed.

Figure 2.5 illustrates the architecture of Maconomy Web Services described above.

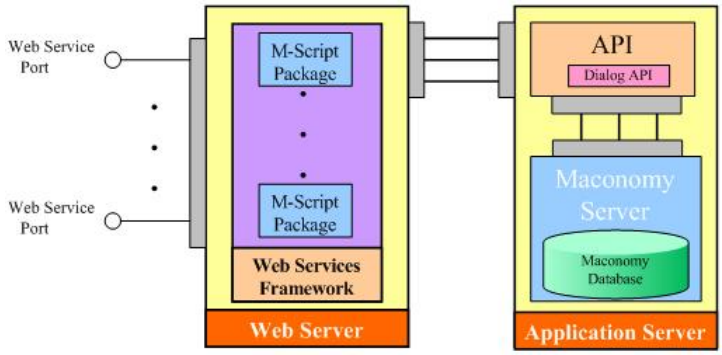


Figure 2.5: The architecture of Maconomy Web Services

For instance, we may create a M-Script file named "sample.ms" in which a few public functions containing the invocations to Maconomy API are defined and the variable **serviceDescription** is also added to describe the service properties. Then we just need to place it in a proper directory on the Maconomy web server. Automatically the Maconomy Web Services Framework will generate the following **Web Service Ports** for this file, so that the service is ready to be used by any Service Consumer.

- WSDL: http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe/wsdl.ms?service=sample
- SOAP: http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe/soap.ms?service=sample
- HTTP GET: http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe/get.ms?service=sample

(Assuming the Web service framework is installed on the web server with the base URL "http://your.host.com/cgi-bin/Maconomy/MaconomyWS.exe".)

2.4 What is a Maconomy Timesheet?

In the Maconomy Business Solutions, a timesheet application is provided as a standard component. It is used for a week by week registration of the working hours of each employee. By making a logon to a Maconomy client, each employee

can open a timesheet dialog with a specified date (by the unit of a week). The typical operations within a timesheet dialog are as follows:

- Create a timesheet, if the timesheet for that week does not exist.
- Add a new timesheet line, in which some fields should be filled, such as the project No., task No., and the number of working hours for each day of a week.
- Modify an existing timesheet line.
- Submit a timesheet.
- Switch to a timesheet of another week.

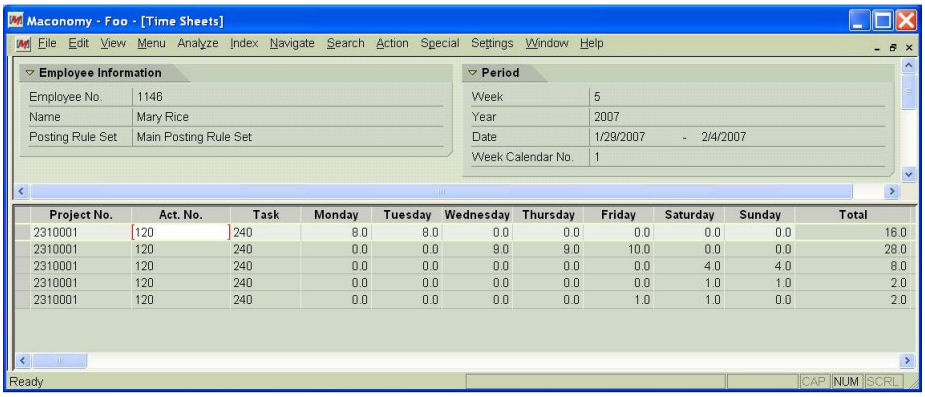


Figure 2.6: Maconomy Time Sheets Dialog

Figure 2.6 is a screenshot of a part of the "Time sheets" dialog within the Maconomy Windows Client. The dialog is divided into two segments. The upper one displays the "Employee Information", and "Period" (the date and the week No.). The lower one displays the registered timesheet lines, which are arranged in a table with columns corresponding to each field of a timesheet line.

Chapter 3

Possible Solutions

In the area of specialized client-server protocols for Service Oriented Architectures, two specific standards were investigated. One is a vendor specific proposal: the Microsoft Office Information Bridge Framework. The other is in the context of web-portals - "Web Services for Remote Portlets". Both of them are analyzed in this chapter.

3.1 Information Bridge Framework

The Information Bridge Framework (IBF) is a set of software components, tools, and prescriptive guidance that enable developers to create solutions that connect the Microsoft Office System as 'SOC' to virtually any enterprise system as 'Service Provider'.

3.1.1 Overview of IBF

The idea of IBF

Naturally you may ask why Microsoft developed IBF to figure such a connection.

Their motivation comes from a study of Microsoft which reveals that "*there is a deep divide between the enterprise system used to collect and manage data and the desktop tools that information workers use to communicate and act on this information.*"[8] This means that many information workers in an enterprise environment have to frequently switch between the Office System and various enterprise applications, using enterprise systems to discover or operate data, and then copying and pasting that data into the Office System in which they can make decisions based on this data and communicate intelligence to others. Switching applications brings serious problems such as inefficiencies, high opportunities for error and so on. (For details, see [8].)

In order to solve these problems, IBF was provided to help developers build 'Information Bridge' between Office System and enterprise systems so that information workers can establish direct references to enterprise business objects within Office documents(e.g. email, Word) to view, retrieve, and act on business information.

The core concept - IBF Metadata

In IBF, accessing enterprise systems is implemented by wrapping them as compatible Web services. When invoking these services from Office-based clients, the Web services themselves are transparent to clients. This is because that IBF provides a middle layer called 'Information Bridge Metadata' that "*aggregates Web services into conceptual business entities, and enables the passage of data from any Web service into and out of any UI element or presentation.*"[9] In other words, metadata abstracts multiple, diverse Web services into a single service layer to which user interface elements (e.g. menus, windows Forms controls) can connect.

Architecture of IBF

After understanding the idea and basic concepts of IBF, let us have a look at the architecture of an IBF solution which is possibly leveraged to link an office-based client to Maconomy system together.

Figure 3.1 presents the components that constitute Information Bridge Framework:

a) An IBF compliant Web service that encapsulates the back-end enterprise system. In this project, it might be the existing Maconomy Web Services. Especially, the compatibility with IBF is needed to be checked.

b) A Metadata repository (Metadata Service) that stores the customized metadata and exposes itself as a Web service that provides access to the Metadata.

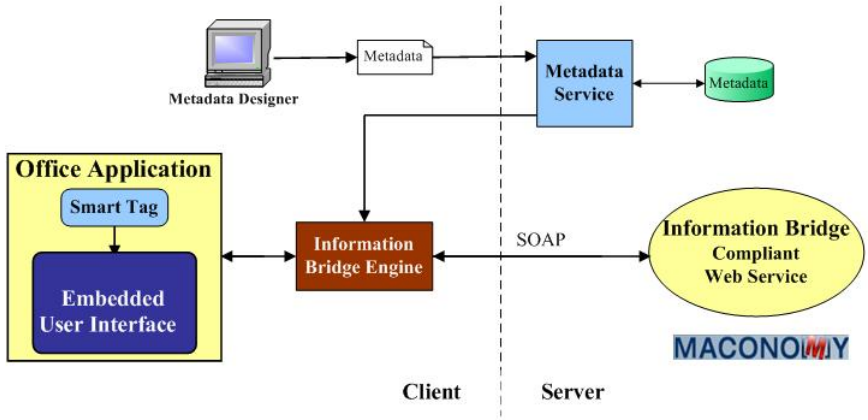


Figure 3.1: Architecture of Information Bridge Solution connecting Office-based UI to Maconomy Web services

The function of Metadata Service is a little like UDDI. Clients will download subsets of this Metadata as needed on basis for execution.

c) IBF Client Engine. This last piece has two distinct components:

1. The Engine which downloads the Metadata from the Metadata Service when needed and keeps a local cache of it. It also understands Metadata and executes it based on the current context. It performs all the non-UI related operations like SOAP calls, transformations, etc. This component is UI agnostic.

2. The UI Engine: "that understands about the application where it's being hosted (Outlook, Word, etc.) and will render the UI and provide services specific to the host application" [10]. It creates an abstraction layer on top of the hosting application so that solutions built with IBF don't need to know about differences between hosting applications.

d) Metadata Designer is a Visual Studio based tool that allows for editing metadata and publishing them to the Metadata Service.

To further validate the possibility of utilizing IBF to connect generic clients to Maconomy Web services, the following three sections analyze IBF in three distinct stages, from the point of view of designing an IBF solution.

3.1.2 Creating an IBF compliant service

From Chapter 2, we can see that Maconomy has established a Web Services Framework to allow the M-Script programmer to expose any M-Script package as a Web service which can provide full access to Maconomy database.

To check if Maconomy Web Service is IBF compliant, we should first get to know the requirement that IBF imposes on web-services.

Normally, Web services expose quite a bit of information about how the service can be consumed but offer very little help in understanding what type of information or functionality is offered. In other words, Web services usually expose WSDL so that tools can easily discover what methods and parameters the Web service exposes but offer little clues about what business entities are defined behind those methods or even if they affect the back-end systems at all (no way to say if a method will update Maconomy database for example). It seems that WSDL is not enough for representing what Web services expose.

Service Metadata

Therefore, IBF proposed "Service Metadata" (the part of Metadata), which not only stores the service definitions for Web services the solution needs but also aggregates data sources and maps them into conceptual business entities, which are named using business terminologies. By this way, the UI developers do not need to know how to interact with web services, but only **link** the user interfaces with the elements defined in the service metadata to provide the data and operations an end user might need. Here, the **link** is defined by the other part of Metadata - "UI Metadata", which will be discussed later. Figure 3.2 illustrates the role the service metadata plays:

In the Service Metadata, the following elements need to be defined:

- **Entities** – "*Abstract business or user definitions that will encapsulate a set of data or functionality.*" [10] For example, we can have an "Employee" entity or the Maconomy timesheet as an entity.
- **Views** – "*A schema associated with an entity that describes a subset of data about it.*" [10] For example, for the Employee entity we may have several views like Employee Contact Information or Employee Competency Information. Each view complies with a particular schema and is a representation of the entity for a given context.
- **Relationships** – Entities/Views can be related to others and these relationships should be described in this Metadata. For example, an Employee

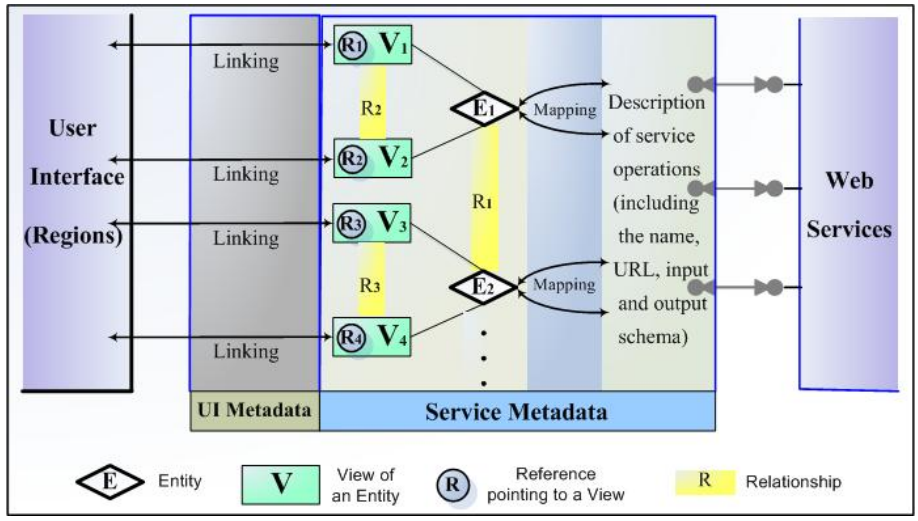


Figure 3.2: The functionality of IBF Metadata

may be related with his Timesheet. "*Relationships allow for navigations between the entities by just executing the Metadata description. A relationship then will describe how to get from one entity into another.*"[10]

- **References** – "*A reference is a common way to point to a set of information. It is a schema and represents the minimum set of information needed to retrieve a piece of data.*" [10], e.g. Employee No. for retrieving an Employee's personal information. There can be multiple ways you can retrieve a piece of information, e.g. an employee could be retrieved by name, Id, CPR No., etc.

Web Service Operation Design

Why did we analyze the IBF service metadata before discussing the creation of an IBF compliant web service? That is because the web services with the specific structure can facilitate the definition of the corresponding service metadata. In other words, it would be easier to use the service metadata to map the data and operations provided by web services into the business entities.

Now we should have a look the Web service design suggested by IBF. Three categories of operations are recommended:

- **Get** – A 'get' method is one that allows you to retrieve the data for an entity/view by passing a Reference. An example would be a method

called 'GetEmployeeContactInformation' that would accept an Employee No. Reference parameter.

- **Put** – This is a method that allows you to modify the content of an entity/view by updating the back-end system. It accepts two inputs, the Reference to the entity/view to update and the data to be updated.

- **Act** – This kind of method allows for doing things that change the state of the system that hosts information. For instance, an operation called 'SubmitTimesheet' that accepts the Reference to a timesheet, then changes the status of a timesheet to 'submitted' and returns the result (success/fail).

By analyzing the function of IBF Service Metadata and the suggested forms for designing Web service operations, it is not difficult to find that the requirement IBF imposes on Web services is exactly the "reference-based" operation structure. Here, the reference defined as the input of an operation should have the same schema as the one to be described in the service metadata.

When creating Maconomy Web services in M-Script, most of the functions are normally designed to invoke API to access data in the Maconomy database. Typically, operating the data in a database often needs a key to locate a set of data. For example, we may have a key composed of an employee number and a date to retrieve or update a record of the table "timesheet". Therefore, we can imagine that it is natural to define a reference schema which includes a key field for accessing the Maconomy database and regard the reference as the input of a service operation, which returns a record identified by a value of that key through invoking the corresponding API.

In this way, we can reasonably believe that it is absolutely possible to design the Maconomy Web services to be of the "reference-based" operation structure and expose them to IBF through Maconomy Web Service Framework.

3.1.3 Building IBF Client User Interface

From Figure 3.1, we can see that Office products are the host applications running at the client side of IBF. The customized user interface designed for various enterprise users could be embedded as a UI component into Office applications. In an IBF-based solution, the data for an entity is displayed in the IBF information window. For example, IBF uses 'Document Actions task pane' in Office Word 2003 and a modeless dialog window in Office Outlook 2003 to serve as information window.

In order to start up the IBF information window within Office applications to display back-end data, the **Office SmartTag** design is extended in IBF. "*IBF allows office documents to contain live links to back-end data. The way these documents contain information about what back-end data to obtain is via SmartTags.*" [10] The live link (SmartTag) is visible to the user as a dotted purple line underlining a term and stores information about what back-end piece of information is pointing to. Essentially, a smart tag is a small block of XML code containing the so-called "Context Information", which includes the name of a specific entity, the name of a specific view within that entity and an instance of the reference to the View. When the IBF client engine receives the Context Information, it will locate the specified View in the metadata and use the Reference instance to retrieve the corresponding view data (back-end data) to be displayed in the UI controls.

A solution needs to define how it wants these SmartTags to get into the document and IBF provides several ways for doing so. Firstly we may manually generate a document with the SmartTags embedded (this would be useful if the emails/documents are dynamically generated by some other processes). The second is to use a SmartTag recognizer to detect pieces of text based on a regular expression or by doing a look up and dynamically insert a SmartTag in them. The third is to use the built in Search capability in IBF for the user to find the instance of information they are interested in and allowing them to paste it into the document.

The IBF information window provides a Window Pane approach to display details of back-end data on one or more regions that are fully definable by the solution developer. IBF supports .NET CLR controls and HTML regions (and menus for those regions). Creating a piece of UI is just a matter of creating a windows GUI control and implementing the existing IBF interface that will get the data into the control. The UI control itself does not need to know how or where the data is coming from. The control only needs to know the type of data that will be provided. IBF will automatically instantiate the control at run time and will pass the right data to the control. This allows a separation of displaying data from retrieving that data, which is solved by defining the UI metadata.

3.1.4 Creating UI Metadata

The IBF Metadata can be divided into two categories: Service Metadata and UI Metadata. Since Service Metadata has been discussed in the first step, the final step for creating an IBF solution is to describe the UI Metadata that will link the business entities/views defined in Service Metadata with the UI elements.

IBF provides a few elements that actually function as the "link":

- **Actions** – "These are the executable units from a user point of view and can contain both Service and UI methods/operations." [10] In the previous 'Employee' example, we would have a DisplayInformation action that would use the "EmployeeContactInformation" view of the "Employee" entity and would link it to a UI control displaying employee information.
- **Transformations** – Because data from the Web service and the data required by the UI elements might not be the same, IBF allows us to transform the data. For example, the complete 'EmployeeProfile' data returned by the Service operation GetEmployeeProfile needs to be transformed to a subset (ContactInformation) of itself which then can be displayed in the "Employee Contact Details Region". XSL transformations, regular expressions or calling CLR components are all supported ways to transform data.

With the understanding of these concepts, we can use Metadata Designer to define the certain metadata which maps the view data into the corresponding UI elements embedded in an Office application.

Summary

Through the above three steps, we can see that Information Bridge Framework could be an option to connect an Office-based generic client/SOC with Maconomy Web services by separating the Service layer from the UI layer and linking them via Information Bridge Metadata.

3.2 Web Services for Remote Portlets

This section analyzes Web Services for Remote Portlets (WSRP), a specification which has been produced through the joint efforts of the Web Services for Interactive Applications (WSIA) and Web Services for Remote Portals (WSRP) OASIS Technical Committees.

3.2.1 What is WSRP?

The WSRP is a specification that defines a common, well-defined interface for communicating with pluggable, Presentation-oriented Web services. In this def-

inition, two important terms need to be explained.

- **Presentation-oriented Web services**

  The Web services are presentation-oriented which means that they provide a user interface that allows an end-user to interact directly with the Web services. This is completely different from the traditional data-oriented Web services which receive requests and return data objects in the response.

- **A common, well-defined interface**

  The common, well-defined interface governs how a portal communicates with the Web services and collects the mark-up fragments it needs to present a page to the end-user. It is precisely this common interface which allows portal applications to generically consume portlets running in remotely-running containers.

Related Standards

WSRP is built upon existing Web services standards like SOAP, WSDL and UDDI. Figure 3.3 shows how WSRP fits into the Web services standards stack.

Figure 3.3: WSRP relies on existing Web services technologies

WSRP uses WSDL to formally describe the WSRP service interfaces, requires SOAP binding be available for invoking WSRP services and optionally uses UDDI to register the exposed WSRP services.

Since WSRP services return mark-up fragments as response to a portal application, WSRP also defines the notion of “valid fragments of markup’ based on the existing markup languages such as HTML, Voice XML, cHTML, etc.

3.2.2 Architecture of WSRP

Within a WSRP architecture, a couple of basic concepts should be clarified.

- **Portal**: "*A portal can be thought of as a Web-based application that is customizable by the end-user both in the look and feel of the portal and in the available content and applications which the portal contains. A portal, furthermore, can be thought of as an aggregator of content and applications or a single point of entry to a user's set of tools and applications.*" [11]
- **Portlet**: "*A portlet can be thought of as a miniature Web application that is running inside of a portal page along side any number of similar entities. A portlet is a Web component which is managed by a container and can process requests and generate dynamic content. Portlets come in many flavors – some are standards-based, while others are proprietary to the portal which hosts them.*" [11]

Figure 3.4 below illustrates each of the primary actors within a WSRP architecture and the role a portal plays in aggregating the mark-up fragments. Although this diagram shows a portal consuming WSRP portlets from only a single producer, there is no reason why a portal couldn't consume portlets from any number of WSRP producers.

The WSRP Specification defines the following actors within a WSRP architecture:

- **WSRP producer**: This is a Web service that offers one or more portlets and implements a set of WSRP interfaces, thus providing a common set of operations for consumers. Depending on the implementation, a producer could offer just one portlet, or could provide a run-time (or a container) for deploying and managing several portlets. The WSRP producer is a true Web service, complete with a WSDL and a set of endpoints. Every producer in WSRP is described using a standardized WSDL document.
- **WSRP portlet**: A WSRP portlet is a pluggable user interface component that lives inside of a WSRP producer and is accessed remotely through the interface defined by that producer. A WSRP portlet is not a Web service in its own right (it cannot be accessed directly, but instead must be accessed through its parent producer).
- **WSRP consumer**: This is a Web service client that invokes producer-offered WSRP Web services and provides an environment for end-users to

interact with portlets offered by one or more such producers. The most common example of a WSRP consumer is a portal.

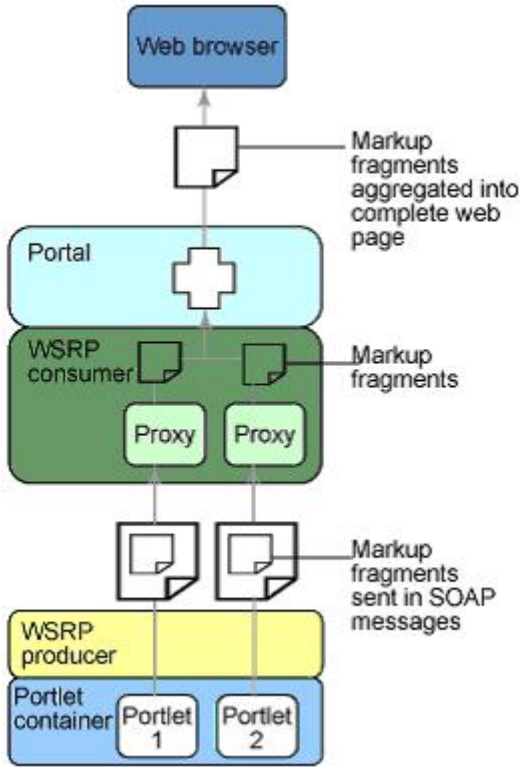


Figure 3.4: "A portal acting as a WSRP Consumer to Aggregate mark-up from remote portlets" [11]

3.2.3 WSRP interfaces

The WSRP specification defines a set of common interfaces that all WSRP Producers are required to implement and which WSRP Consumers must use to interact with remotely-running portlets. These standard interfaces allow a portal to interact with WSRP compliant Web services generically, with the exact same signatures of service operations and the common data structures/types. The WSRP Specification requires that every producer implements two **required** interfaces, while allowing to **optionally** implement two others as well:

- **Service Description Interface (required)**

  The Service Description Interface allows a WSRP producer to advertise its services to perspective consumers. A WSRP consumer can use this interface to query a producer to discover what portlets the producer offers, as well as additional metadata about the producer itself. This interface can not only act as a discovery mechanism (like UDDI) to determine the set of offered portlets, but also importantly allows consumers to get metadata about the producer's technical capabilities. The producer's metadata might include information about whether the producer requires registration or cookie initialization before a consumer can interact with any of the portlets.

- **Mark-up Interface (required)**

  The Markup Interface allows a WSRP consumer to interact with a remotely running portlet on a WSRP producer. For example, a consumer would use this interface to first obtain the markup of a portlet or perform some interactions when an end-user submits a form from the portal page. Additionally, a portal might need to simply obtain the latest mark-up based on the current state of the portlet (for example when the user clicks refresh or interaction with another portlet on the same page takes place).

- **Registration Interface (optional)**

  The Registration Interface allows a WSRP producer to require that WSRP consumers perform some sort of registration before they can interact with the service through the Service Description and Mark-up interfaces. Through this mechanism a producer can customize its behavior to a specific type of consumer. For example, a producer might filter the set of offered portlets based on a particular consumer. In addition, the Registration Interface serves as a mechanism to allow the producer and consumer to open a dialogue so that they can exchange information about each others' technical capabilities.

- **Portlet Management Interface (optional)**

  The Portlet Management Interface gives the WSRP consumer access to the life cycle of the remotely-running portlet. A consumer would have the ability to customize a portlet's behavior or even destroy an instance of a remotely-running portlet using this interface.

3.2.4 Markup Fragment Definitions

In addition to a web services interface and protocol, WSRP also standardizes markup fragments returned by WSRP services for the relevant markup lan-

guages including HTML, XHTML, WML, VoiceXML and cHTML. This definition specifies the markup fragment in the four main aspects listed below:

- The tags which a portlet must not use in the specific markup language
- The format (encoding) of the portlet URLs embedded in a markup fragment
- Namespace schema used to differentiate the names of various elements which are aggregated into one portal page from different portlets
- The common CSS style which all portlets aggregated in one page need to use

3.2.5 Implementing WSRP services with M-Script

Through the investigation on the WSRP specification above, we may conclude that any Web programming language can be used to implement the WSRP compliant Web services by defining the data types and operations that implement the standard WSRP interfaces (at least, the Service Description Interface and Markup Interface). When multiple portlets are to be aggregated in one portal page, the rules for markup fragment definitions must be conformed.

Since the Maconomy M-Script language is designed for generating dynamic web pages in HTML based on data from the Maconomy database and it is in many ways similar to other Web programming languages such as JavaScript, it is reasonable to believe that M-Script has the capability to define the standard data types and operations that the WSRP specification requires and generate the markup fragments that conform to the rules for defining markup fragments.

Summary

Based on the investigation, we can see that the advantage of WSRP is the capability of aggregating multiple portlets from the different services into one page and the high efficiency and flexibility by using presentation-oriented Web services. However, this feature is currently not so meaningful to 'Maconomy - the single data source' that the further experiment on implementing WSRP services with M-Script is not to be performed in this project.

Chapter 4

Implementation of an IBF-based prototype

Based on the analysis to the possibility of leveraging IBF or WSRP to access Maconomy system from a client, a prototype which can expose Maconomy 'Timesheet' functionality to users through IBF has been proposed and implemented successfully to demonstrate the viability. In this chapter, I refer to this prototype as the IBF-based Timesheet. (Please refer to Section 2.4 for an introduction of Maconomy Timesheet Dialog.)

This chapter starts with specifying the prototype functionality, then describes the practical development process and finally gives a discussion.

4.1 The functionality of the IBF-based Timesheet

Firstly you may ask why the prototype was proposed to expose the Maconomy Timesheet functionality, but not another one. The proposal was made with the consideration that Timesheet is traditionally one of the frequently used MScript components in Maconomy, because 90% of all active users do enter time sheets

and most companies which are Maconomy's customers do have a deadline when all time sheets must be finalized.

To define the functionality of the IBF-based Timesheet, a typical use case scenario for the prototype is described as below:

1. User opens a Word document or Outlook email message containing the character string 'timesheet'.
2. When the user hovers the mouse over "timesheet", a smart tag icon appears.
3. User clicks the "Show Timesheets" menu item.
4. IBF task pane titled "Maconomy Timesheet" appears containing a Region that exposes "Maconomy Access Control" interface to users.
5. When the user inputs the username and password, the "OK" button is enabled to let the user click it and then access Maconomy system.
6. If the provided username and password are correct, the user's timesheet for the current week is displayed in a new Region.
7. Now the user can view the timesheet data or switch to another timesheet by selecting a date and clicking the button "Display".
8. When the user clicks the button "Add Timesheet Line", a modal dialog box pops up to ask the user to input project no., task name, etc. By clicking the button "Submit" in the modal dialog box, the dialog box is closed and a new timesheet line is added into the current timesheet.
9. Returning to the timesheet Region, users need to click the button "Display" to update the display of the modified timesheet.
10. If the button "Submit timesheet" is available, users can click it to submit the current timesheet. Otherwise the disabled button means this timesheet has been submitted with the latest modification.

Note: In the above definition, we need to notice that the operation - "Create a timesheet" is not supported in the IBF-based Timesheet solution. This point is different from the existing Maconomy Timesheet application. That means all of the timesheets which are going to be viewed in the prototype solution have to be created in advance through one of the established Maconomy clients.

With the explicit definition to the functionality of the IBF-based Timesheet, now we can go on with the practical development for the prototype.

4.2 Setting up the Development Environment

The implementation of the prototype starts with setting up the development environment. As analyzed in Chapter 3, the development of an Information Bridge solution can be divided into three sequential phases:

1. Web Services Development: The creation of a Web service, which provides the SOAP methods for retrieving and modifying the timesheet data.
2. Metadata Configuration : The definition of Service Metadata and Solution Metadata, both of which play the role of mapping the data source and Web service operations to the Information Bridge terms (Entity, View, Reference and Action, etc) that constitute the context information to be referenced by the UI components.
3. Building UI Components and Smart Tags: The development of Office-based user interfaces and Office smart tags. The former renders the timesheet information to end users and the latter makes end users access the IBF solution.

During the implementation, I achieved the above three phases on two computers running Microsoft Windows Server 2003 as illustrated in Figure 4.1

The phase 1 was performed on PC1 with the task of defining the Web service under Maconomy Web Service Framework. The phase 2 and 3 happened on PC2 with the Visual Studio .NET and a Metadata Designer Plug-in as the main development tools.

To prepare for the development on PC1 and PC2, the following system components and tools are prerequisite and need to be installed respectively on the two machines.

At the side of Web services - PC 1

- Install Maconomy X SP3 with “demo data” using “MConfig”. The instruction for Maconomy installation could be gotten from Maconomy’s “MConfig User Manual”.
- Configure the local machine (Computer 1) as the web server for the current installation. In the page for setting Web Server, we need to make sure the option for ‘Web Services’ is checked.

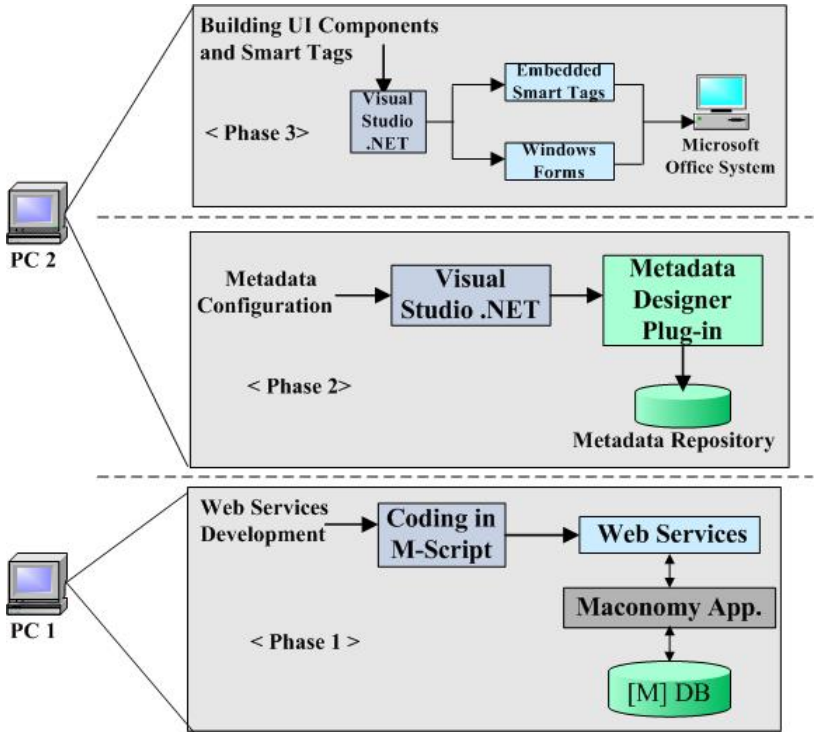


Figure 4.1: Development phases of the IBF-based prototype

- Find the directory <root>/MaconomyWS/Services/Custom. Here <root> is the root of the web service installation. It is just this directory that M-Script packages used to define Web services reside.
- Download the Web Service testing tool - "WebServiceStudio 2.0" from http://www.gotdotnet.com/. It is an executable which can be run directly on Windows.

At the side of Information Bridge Development - PC 2

- Install Internet Information Services (IIS)

  IIS is utilized by Information Bridge Framework Service to expose a pair of Web services used to read and write the metadata from Metadata Repository.
- Install Visual Studio .NET 2003 Professional and .NET Framework 1.1 SP1

Visual Studio works as the main development tool for building Metadata, UI controls and IBF Smart Tag.

- Install and Configure Microsoft Office Professional Edition 2003

  During the installation of Office applications, we need to make sure that the installation options - '.NET Programmability Support' for each IBF-enabled Office application and 'Smart Tag .NET Programmability Support' for 'Office Tools' are selected.

- Install SQL Server 2000 SP3a

  SQL Server works as Metadata Repository to store the solution metadata.

- Install Information Bridge Framework Service 1.0

  IBF Service is the web service through which IBF clients retrieve metadata and Metadata Designer publish metadata.

- Install IBF 1.5 Client Component

  IBF client-side component interprets the context of a region and then connects the context to relevant actions.

- Install MSXML 4.0 SP2

  It is required by the Metadata Designer Tool.

- Install Information Bridge Metadata Designer 1.5

  A plug-in for the Visual Studio .NET is needed for creating and managing solution metadata.

4.3 Creating an IBF compliant Web Service

Since the requirement IBF imposes on Web services is the reference-based operation structure (we got the conclusion from Chapter 3), it is a good idea to start with designing the overall Web service structure in terms of entities, views and operations. The normal way to do this is to identify these Information Bridge concepts from the users' requirement, which is the use case scenario in this case.

◇ Identifying the entity, reference, view and operations

- **Entity**

  Since the prototype exposes the functionality of a Maconomy timesheet, we may just define the business object - "timesheet" as an entity.

- **View**

  Each timesheet would have some data (e.g. timesheet lines) associated with it. We should define a view named "TimesheetDetails".

- **Reference**

  Since an Employee Number and a definite date are combined to locate a unique timesheet, we should define them as the reference to the entity "timesheet".

- **Operations**

  From the use case scenario, we can see that the prototype provides:

  – A 'Get' operation to allow the user to retrieve the timesheet data,

  – A 'Put' operation to allow the user to register new timesheet lines,

  – An 'Act' operation to allow the user to submit a timesheet.

With the overall structure designed, now we can start creating the Web service.

♢ Defining data types and methods in M-Script

Maconomy Web services Framework allows the use of M-Script sessions in the services and the session ID is automatically transported in the SOAP header. Therefore, we have the choice to determine if an operation requires a session ID. But considering the complexity of handling the data in the SOAP header within IBF, I chose to define the IBF compliant web service without using session ID. This leads to the fact that any operation containing the invocation of Maconomy API has to log onto Maconomy system first and log out finally.

Before defining the web service operations, we need to define data types for the input and output of the operations which are type of Get or Put or Act.

- **referenceType**

```
var referenceType =
{
  type: "object",
  elements:
  {
    username: "string",
    password: "string",
    PeriodStart: "date"
  }
};
```

'username' and 'password' are used to log onto Maconomy system and then the employee number of the user can be gotten by invoking Maconomy SQL API. A specified timesheet is located by the values of "Employee Number" and "PeriodStart".

- **timesheetType**

```
var timesheetType =
{
  type: "object",
  elements:
  {
    EmployeeNumber: "string",
    PeriodStart: "date",
    Approved: "bool",
    Rows:
    {
      type: "array",
      elements:
      {
        type: "object",
        elements:
        {
          JobNumber: "string",
          ActivityNumber: "string",
          TaskName: "string",
          NumberOfDay1: "real",
          NumberOfDay2: "real",
          NumberOfDay3: "real",
          NumberOfDay4: "real",
          NumberOfDay5: "real",
          NumberOfDay6: "real",
          NumberOfDay7: "real"
        }
      }
    }
  }
};
```

This type defines the content of a specified timesheet which is returned by the operation - "getTimesheet".

- **tsLineType**

```
var tsLineType =
{
  type: "object",
  elements:
  {
    JobNumber:      "string",
    ActivityNumber: "string",
    TaskName:       "string",
    NumberOfDay1:   "real",
    NumberOfDay2:   "real",
    NumberOfDay3:   "real",
    NumberOfDay4:   "real",
    NumberOfDay5:   "real",
    NumberOfDay6:   "real",
    NumberOfDay7:   "real"
  }
};
```

"tsLineType" defines the content of a timesheet line which is one of the input parameters for the operation - "register".

- **The signature of Web service methods in M-Script**

  1. function getTimesheet(refer)
     Input: an object of type "referenceType".
     Output: an object of type "timesheetType".
  2. function register(refer,entry)
     Input: an object of type "referenceType",
     an object of type "tsLineType".
     Output: true or false.
  3. function submit(refer)
     Input: an object of type "referenceType",
     Output: true or false.

4.4 Testing the web services

By using the web service testing tool - **WebServiceStudio 2.0**, the web service operations designed for reading and writing Maconomy timesheet data are tested as below:

As shown in Figure 4.2, the test tool needs a user to provide the URL of the WSDL endpoint of the web service to be tested. When the user selected the "getTimesheet" operation, the test tool asked the user to input the required parameters. After invoking the operation, the output of that operation is displayed in the output window.

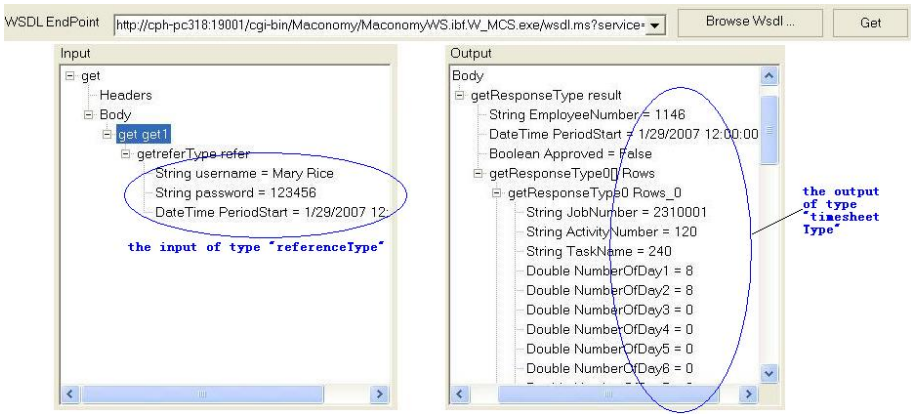


Figure 4.2: The test to the 'getTimesheet' operation

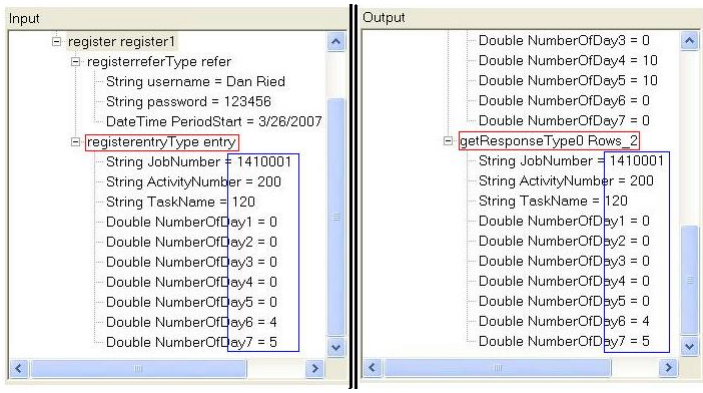


Figure 4.3: The test to the 'register' operation

In Figure 4.3, the left window is the input window for the operation "register" and the right window is the output window for the operation "getTimesheet" which was invoked after calling "register" to add a new timesheet line. From the figure, we can see that the input timesheet line was added to the specified timesheet.

4.5 Describing IBF Service Metadata

After creating the web service, Maconomy Web Services Framework can automatically generate the service WSDL. Based on the WSDL, we can define the service metadata through Metadata Designer. The steps for describing the Web Service functionality in metadata are listed below:

1. Create a metadata project named "TimeSheetMetadata" and then import the WSDL of the created web service to generate the base service metadata.
2. Assign the value "Maconomy.SampleSolution.TS" to the Name property of the Metadata scope and the value "TimeSheet" to the DisplayName property.
3. Add an entity named "TimeSheet" and add a view named "TimeSheetDetails" under the node of "TimeSheet".
4. Create schema elements corresponding to the input and output types of the web service operations and specify the port property of each schema element with the same value "Xsd.maconomy.com" which is the existing port element added by the import tool.
5. With the schema elements created above, assign the schema name "getResponse (maconomy.com)" to the Schema property of the view "TimeSheetDetails".
6. Add operation elements corresponding to the web service operations and assign values to the following properties of each operation element:
   - **Entity**: select the value "TimeSheet" from the drop-down list.
   - **Type**: "OperationSoapRequest"
   - **Input & Output Schema**: select the corresponding value from the list of existing schemas.
   - **Port**: assign the value "Soap.timesheetSOAPPort" which is the existing port element added by the import tool.
   - **SOAP Action**: specify with the value "timesheet".

With these changes, the metadata now contains all the artifacts required to describe the Web Service functionality to the IBF client applications. The XML file created in this process can be viewed by opening MSIBFMetadata.xml.

4.6 Building Smart Tag & Custom Action Handlers

In this prototype, Smart Tag was selected as the entry point to access the Timesheet IBF solution. IBF allows developer to optionally register one or more recognizers and custom action handlers in metadata. Recognizers identify keywords in user content and insert smart tags into the user's document. Action handlers manage smart tag menu commands. With a custom action handler, we can choose a customized menu item to show the user and define what actions to perform in response to user selection.

Since the word 'timesheet' is the only text used as a smart tag to access the Timesheet solution, I chose to manually insert a smart tag to identify the word 'timesheet' but not register any recognizer into metadata. However, a custom action handler was built to show "Show Timesheets" menu item to the user and perform the action to display the "Maconomy Access Control" region in the task pane according to the 'Context Information' contained in the smart tag.

♢ **The context information contained in the smart tag** which could be inserted into an Office document is shown below:

```
<?xml version="1.0"?>
<ContextInformation
  MetadataScopeName="Maconomy.SampleSolution.TS"
  EntityName="TimeSheet" ViewName="LogonMaconomy"
  ReferenceSchemaName="AccessSchema"
xmlns="http://schemas.microsoft.com/InformationBridge/2004/ContextInformation">
  <Reference>
    <Dialog Name="Timesheet" xmlns="maconomy.com" />
  </Reference>
</ContextInformation>
```

♢ **Coding a Custom Action Handler**

Information Bridge registers one generalized action handler with Office and then forwards all calls reaching this generalized action handler to action handlers specified in the solution metadata. We call an assembly which contains the classes of recognizers or action handlers as a smart tag assembly.

To create a smart tag assembly, I created a new class library in Visual Studio and add a reference to the .NET assemblies:
Microsoft.InformationBridge.Framework.Interfaces.dll,
Microsoft.InformationBridge.Framework.UI.Interop.dll, and
Microsoft Smart Tags 2.0 Type Library.

Next, the steps are to create a class named "TimeSheetActionHandler" which implements IActionHandler .NET interface and implement the members of the interface. Two core member methods of **IActionHandler** interface are:

- The method **GetMenuItems** returns an array of valid menu items

  IActionHandlerMenuItem[] GetMenuItems(IMediator, ISmartTagProperties)

- The method **InvokeMenuItem** executes an Action against IBF based on which menuItem is passed in.

  void InvokeMenuItem(IActionHandlerMenuItem,IMediator, IContextInformation)

♢ **Referencing Smart Tag Assembly in Metadata**

Recognizers and custom action handlers are managed within metadata. The IBF Resource Kit includes starter metadata that provides entities we can use to add a custom action handler to the solution. The main steps involved are:

- Import starter smart tag metadata from the template file in IBF Resource Kit.
- Modify starter smart tag metadata to point to the smart tag assembly we created.
- Define the properties of "GenericActionHandler" and change its "TypeName" value to the namespace and name of the class **TimeSheetActionHandler**.
- Load the custom action handler by adding the operation "LoadActionHandler" into the action contained in the view "GenericActionHandlers".

4.7 Building User Interface Components

To make users interact with the IBF-based Timesheet, I totally created three IBF regions which work as the user interface respectively for "Maconomy Access Control", "Display Timesheet Data" and "Register a timesheet line".

Before getting into the implementation detail of each region, it is good to know the common interface which these IBF regions extend. For each specific region, I mainly describe the aspects related to the execution of commands against IBF.

♢ The Common Interface

First I started by creating a Windows Control Library project in Visual Studio and then add project references to two IBF interfaces:
Microsoft.InformationBridge.Framework.Interfaces.dll,
Microsoft.InformationBridge.Framework.UI.Interop.dll,
which are installed in the global assembly cache (GAC) during an Information Bridge Framework client installation.

By adding their namespaces to the source file of each region class, each region can implement the IRegion interface. It is through **IRegion** that the user controls will be able to get data, styles, and task pane frame information passed to them from Information Bridge Framework. Two required property overrides for IRegion are defined in the implementation of each region:

- **Data Property**

  It is used to set the data returned by the Information Bridge Framework engine (e.g. View data), and is of an XmlNode data type - a class defined in .NET Framework class library. It is perhaps the most important of the IRegion members. The following code represents the Data property:

```
public XmlNode Data
{
    set
    {
        XmlNode dataNode = value;
        //Call the LoadXmlData routine to insert data in UI controls
        LoadXmlData(dataNode);
    }
}
```

  If a region is designed to render View data to users, we need to process the **XmlNode** to extract the data and then place it in the corresponding controls. An instance of the function LoadXmlData() is described in the part of "DisplayTimesheet" Region.

- **HostProxy Property**

  HostProxy is used to get a reference to the object **IRegionFrame**. The frame object makes us perform any actions such as executing an Information Bridge Framework command or adding a region menu. The following code initializes HostProxy:

  When we have a HostProxy reference, we may use its Host property to return its IRegionFrame object. IRegionFrame essentially acts like the "Application object" of the IBF solution, and it is the jumping off point

```
public IRegionFrameProxy HostProxy
{
    set
    {
        this.regionFrameProxy = value;
    }

}
```

for access to several other objects, including Mediator and RegionMenu. In our prototype solution, **Mediator** is often used to execute an IBF command as shown later.

By inheriting the above interface, each 'Region' object has the capability to interact with IBF. Next, we should get to add windows GUI controls for each region and define the corresponding event handlers for the possible user interaction events.

♢ "LogOn" Region

The main functionality of "LogOn" region is to provide the UI through which users can get access to Maconomy system by inputting the registered username and password. When the user clicks the 'OK' button, a command will be executed against IBF to get the timesheet data of the current week for the user. The following code is the event handler for clicking the 'OK' button.

```
// Click event when a user clicks OK button. This will invoke Information
// Bridge to bring up the timesheet data of the current week.
private void button1_Click(object sender, System.EventArgs e)
{
    //Preparing the context information for the command
    Constants.user = username.Text;
    Constants.code = passwd.Text;
    Constants.selectedDate = DateTime.Today;
    String currentdate = Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-dd");
    string formattedString = string.Format(
        CultureInfo.InvariantCulture,
        Constants.TimeSheetContextXmlFormatString,
        Constants.user, Constants.code, currentdate);
    IContextInformation contextInformation =
        Facade.ContextFactory.DeserializeContextInformation(formattedString);

    IEngineCommand command = regionFrameProxy.Host.Mediator.CreateLocalCommand(
        "MSIBF.UI.ShowContext",
        contextInformation.Reference);
    command.Context = contextInformation;
    command.Execute();
}
```

As shown in the above source code, the variable named "formattedString" is used to compose a piece of IBF context information in which we can appoint an entity, a view of the entity and set an instance for the reference of the view. With the context information as arguments, we can use the method "CreateLocalCommand" of the object - **Mediator** (mentioned in the common interface) to create a command. The execution of this command actually runs a special action within the appointed view of type **EnterContext**. In this case, the action of type EnterContext defined in Metadata will show the "DisplayTimesheet" region to users, with the timesheet data of the current week displayed in it.

♢ "DisplayTimesheet" Region

When a user opens this region, the timesheet for the current week will be displayed. The operation offered to a user by this region is to select a new date and then click the 'Display' button to show a new timesheet or click the 'Add Time Sheet Line' button to open the 'RegisterLine' region for adding a new timesheet line. The event handler for clicking the 'Display' button has the similar code as the handler for clicking the button 'OK' in the "LogOn" Region. The following code is the event handler for clicking the 'Add Time Sheet Line' button, which performs an IBF action to open the 'RegisterLine' region.

```
// Perfom an IBF action to open the RegisterLine region
private void button1_Click(object sender, System.EventArgs e)
{
    IEngineCommand command =
        regionFrameProxy.Host.Mediator.CreateActionByNameCommand(
        Constants.ShowAddTimesheetLineActionName,null);
    command.Context.MetadataScopeName = "Maconomy.SampleSolution.TS";
    command.Context.EntityName = "TimeSheet";
    command.Context.ViewName = "TimeSheetDetails";

    command.Execute();
}
```

In the above code, a command is created to invoke a named action within the appointed view, which is located through defining the three attributes of the command's context: 'MetadataScopeName', 'EntityName' and 'ViewName'.

In the "DisplayTimesheet" Region, another point we need to notice is that this region renders the timesheet data to users. Therefore, we need to provide the implementation for the function **LoadXmlData** referenced in the definition of Data property. Data property is set with the XML data returned by the IBF engine and it is here that we extract the timesheet data and give a call to LoadXmlData to place these data into the UI controls. The following code is a snippet of the implementation for our LoadXmlData function.

```
private void LoadXmlData(XmlNode data)
{
  if (data["EmployeeNumber"] != null )
  {
    emno.Text = data["EmployeeNumber"].InnerText;

  }
        ......
}
```

♢ **"RegisterLine" Region**

This region offers a user to input the value for each field of a timesheet line. When the user clicks the 'Cancel' button, the region will just close itself without any other action. If the user clicks the 'Submit' button, the event handler will collect the values input by the user and use it as the parameters to execute an IBF command which adds a timesheet line into the current timesheet.

```
//Click event when a user clicks 'Submit'. This will invoke Information Bridge
//to add a timesheet line into the current timesheet with values the user input.
private void ok_Click(object sender, System.EventArgs e)
{
    //Construct the parameter object for the command to be created
    string date = Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-dd");
    string actionParameters = String.Format(
        CultureInfo.InvariantCulture,
        Constants.RegisterParametersFormatString,
        Constants.user, Constants.code, date,
        projectNo.Text, actNo.Text, task.Text,
        monday.Text, tues.Text, weds.Text,
        thur.Text, fri.Text, sat.Text, sunday.Text);

    IEngineCommand command =
        regionFrameProxy.Host.Mediator.CreateActionByNameCommand(
        Constants.SubmitTimesheetLineActionName, null);
    command.Context.MetadataScopeName = "Maconomy.SampleSolution.TS";
    command.Context.EntityName = "TimeSheet";
    command.Context.ViewName = "TimeSheetDetails";
    command.Context.Parameters = actionParameters;

    //For error handling
    errorOccurred = false;
    command.Error += new ErrorEventHandler(command_Error);
    command.Execution += new ExecutionEventHandler(command_Execution);
    command.Execute();
}
```

The code shown above has almost the same behaviors as the former event handler for opening the "RegisterLine" region. But the difference is that the command created in this handler is defined with one more attribute named 'Parameters'. Its value is assigned through the variable "actionParameters", which contains the data of the timesheet line a user just added. The execution of this command actually runs an action named 'Submit Timesheet Line' defined in metadata, with the result that a new timesheet line is registered into the current timesheet.

After implementing the above three regions, a successful build in Visual Studio will generate a UI assembly (a dll file), which will be registered into the IBF solution metadata later.

4.8 Describing IBF Solution Metadata

To complete metadata, we still need to add information about how User Interfaces are structured and what actions are provided to the end user by using the Metadata Designer tool. Within IBF, displaying a user interface can be

reached by just navigating to a view in which an action of type **EnterContext** is defined to show the user interface (a region) or by explicitly invoking a **non-EnterContext** action of a view to show it. For the three regions created in this solution, the following metadata elements were added respectively for each of them.

♢ The "LogOn" UI

To structure the "LogOn" UI into the metadata, the required metadata elements should be added and associated like this:

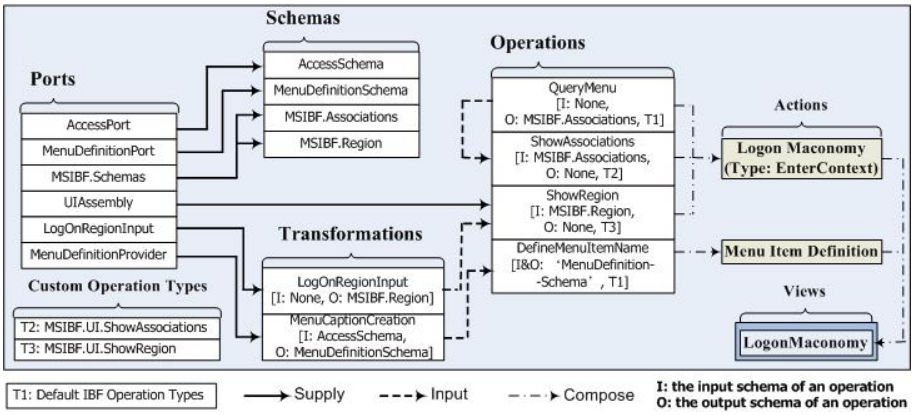


Figure 4.4: Add metadata elements for the "LogOn" UI

As shown in Figure 4.4, the end point - "LogonMaconomy" is a view element used to provide the end user an action of type **EnterContext**, which can be invoked to show the "LogOn" region and a custom menu item in the Office task pane. In order to define the higher level elements, the Ports, Schemas, Transformations, Operations and Actions should be added and associated in sequence so that the top level element - the view "LogonMaconomy" is composed.

- **Ports**: Each port stores the XML or XSLT definition for the corresponding Schema and Transformation element as shown in the figure. The text in each item is the name of that Port element. The detailed definition for each port can be found in Appendix D.
- **Schemas**: Each schema defines a data type which can be referenced as the input or output type of an Operation element or a Transformation element.
- **Transformations**: The "LogOnRegionInput" transformation actually refers to a XML file/port to define the Region properties (e.g. the caption

name, the class name of the User Control.) which is used as the input of the "ShowRegion" operation. The "MenuCaptionCreation" transforms the Reference data got from the SmartTag to XML that describes the caption and tooltip for a menu item.

- **Custom Operation Types**: IBF provides some optional operation types which are defined in Microsoft.InformationBridge.Framework.UI.dll. For instance, **MSIBF.UI.ShowAssociations** corresponds to an operation that internally calls a routine to display the menu items and **MSIBF.UI. ShowRegion** for rendering the User Control (Region).
- **Operations**: In this case, the added operations are used respectively for defining a menu item, querying the menu item definition for the properties of the menu item, displaying the menu item and the UI Controls to the end user.
- **Actions**: The action "Logon Maconomy" is defined to run the operations - QueryMenu, ShowAssociations and ShowRegion in sequence. It is just the only action offered to the end user by the view "LogonMaconomy".

As illustrated in the figure, there are three kinds of relations between the elements.

- **Supply**: The starting point provides the definition for the Port property of the end point.
- **Input**: The output of the starting point is provided as the input of the end point.
- **Compose**: The starting point is a member of the end point.

♢ The "DisplayTimesheet" UI

In order to provide an action which can be invoked to display the "DisplayTimesheet" Region containing the specified timesheet data to the end user, the corresponding metadata elements should be added like what is described for the "LogOn" UI. In this case, we focus on the required top level element - Actions.

In the view "TimeSheetDetails" (which was added when we described the service metadata), the action named "Display Timesheet Details" (of type **EnterContext**) was defined to contain only one operation - "ShowRegion". The Port property of the "ShowRegion" operation is supplied by the Port element UIAssembly, which points to the assembly (the compiled dll file) we created for

our UI. In this manner the IBF engine will understand that Assembly associated with the port UIAssembly needs to be used in the ShowRegion operation to render the output. With the UI Control available, a **View Locator** element is also needed to locate the View Data (the timesheet data) that is to be displayed in the UI Control. Hence, the Web Service operation "getTimesheet" is referenced in the View Locator element to provide the timesheet data as View Data.

When the end user navigates to the "TimeSheetDetails" view, the view locator will automatically retrieve the timesheet data from the Web service and the action of type **EnterContext** will be invoked automatically to render the timesheet data in the corresponding region to the end user.

♢ **The "RegisterLine" UI**

According to the requirement, the "RegisterLine" UI should be displayed in a modal dialog. Hence, another IBF Custom Operation Type -**MSIBF.UI. ShowDialog** was added.

To show the "RegisterLine" region to the end user, we just need to do the similar steps as the former one within the view "TimeSheetDetails". The difference is that we need to define an operation of type **MSIBF.UI.ShowDialog** and add it into an action which is not of type **EnterContext**. The reason is that the event handler will explicitly invoke this action to show the "RegisterLine" region in a modal dialog, when the end user clicks the button - 'Add Time Sheet Line'.

In addition, another action needs to be provided to the end user for registering the timesheet line the user just input into the current timesheet. To do this, an action named "Submit Timesheet Line" was defined to include the Web Service operation "register". This action can be invoked explicitly by an event handler in the "RegisterLine" UI.

For further details about the solution metadata, please refer to the **Appendix D**.

4.9 Testing the solution

To prepare for the solution test, several configurations are needed to be done even in the development environment.

1. Open IIS to make sure the Information Bridge Read and Write services are running.

2. Create an encryption key file by using an executable - sn.exe, a Visual Studio tool. And then configure to trust all of the created assemblies in Microsoft .NET Framework 1.1 Configuration.

3. Publish the metadata into Metadata Service by using Metadata Designer tool.

4. Open the Information Bridge Setting Manager to uncheck the option 'caching metadata in local machine'.

5. Create a Word document to enter the text "timesheet" and save it as .XML file. And then insert a block of context information to mark up "timesheet" as a smart tag. The created XML file can be found in **Appendix B** and the lines in the ***bold italics*** within it are the added context information used to invoke the IBF-based Timesheet solution.

After performing the above steps, now we can test the IBF-based timesheet solution. The screenshots of the testing are attached in **Appendix A**.

4.10 Possible improvements

The IBF-based prototype just implemented the basic functionality of the Maconomy Timesheet, since the focus is to verify the viability of using IBF to connect a generic client with Maconomy Web Services. However, a few aspects are still not full-functioned and are possible to be improved in the future.

• SmartTag Recognizer

In the current prototype, a smart tag around the text "timesheet" is hard coded into an Office document for accessing the timesheet solution as stated in section 4.6. This way is enough for testing the prototype. However, it would be more convenient for the end user to start up the prototype, if creating a smart tag recognizer and referencing it in Metadata. With the recognizer added to identify the keyword "timesheet", a smart tag will be automatically inserted to the document in which an end user is entering the text containing such a keyword.

• The function of creating a new timesheet

Since the function of creating a new timesheet is not supported in the current prototype, an end user has to create the timesheet for a certain week in one of the other Maconomy clients before browsing or editing it in the prototype. Although

this function is not a necessary feature to illustrate the proof of applying IBF, implementing it will make the timesheet solution more functional.

- **The use of session ID**

When designing the web services used for the prototype, I defined the service operations without requiring sessions as stated in section 4.3. The reason is that I am still not very clear about how to configure the IBF metadata to fill the "session ID" into the SOAP request header for invoking a service operation (The session ID is automatically transported in the SOAP header within Maconomy Web Services Framework, but only if the service operation requires it.). Because a login session must be established before being able to access any other function in the M-Script API, each operation in the current service has to first invoke the "login session" API before calling any other API and end with invoking the "logout session" API, so that the "session ID" is not required by any other operation.

Therefore, the possible improvement is to reduce the number of invoking the "login session" and "logout session" API to get the better performance by setting the service operations to use "session ID". Logically, the prerequisite to do this is to make clear how to configure the SOAP request header of a service operation described in the IBF metadata.

Chapter 5

Conclusion

The main objective of this master thesis has been to investigate how Maconomy Web services can support a generic client by leveraging any specialized client-server protocols for SOA. The following activities have been done in this master thesis project.

At the beginning, the Web services infrastructure and the related technologies have been learned and presented. Furthermore, the Maconomy Web Services Framework and required knowledge for defining service operations in M-Script have been analyzed and discussed. All of these were used as the foundation for further analysis, design and implementation.

The overall challenge of this master thesis project has been to investigate two specialized standards for building SOCs and evaluate whether the Maconomy Web Services are compliant to them (IBF and WSRP). In the analysis to IBF, three main aspects for building an Information Bridge solution have been anatomized and a conclusion was drawn that IBF could be an option to connect an Office-based generic client/SOC with Maconomy Web services. For WSRP, the conceps, architecture and the requirement to web services have also been discussed, and it has been found that Maconomy Web Services are capable of generating the markup fragments required by the WSRP interfaces.

Based on the investigation and analysis conducted, a prototype has been pro-

posed and implemented. The prototype was developed based on IBF and exposes the functionality of the Maconomy Timesheet to the Office users. Through the integration testing to the prototype, it was verified that the implemented timesheet solution had achieved the desired functions, so that the viability of leveraging IBF to support a generic client with Maconomy Web services was demonstrated successfully. However, the prototype still needs to be improved in the future work to become fully stable and suitable for the practical application.

Bibliography

[1] "Web Services and Service-Oriented Architectures",
Barry & Associates, Inc
URL: http://www.service-architecture.com/

[2] Eric Newcomer "Understanding Web Services"
Copyright 2002, Addison Wesley Professional

[3] "W3C WSDL 1.1 Specification", World Wide Web Consortium
URL: http://www.w3.org/TR/wsdl

[4] "W3C SOAP 1.2 Specification", World Wide Web Consortium
URL: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[5] Jeff Whatcott, "SOA's Next Wave: Service-Oriented Clients"
URL: http://www.cio.com/weighin/column.html?CID=21201

[6] "M-Script Language Reference", Maconomy Corporation
Internal Documentation

[7] "M-Script Maconomy API Reference", Maconomy Corporation
Internal Documentation

[8] "Introducing the Microsoft Office Information Bridge Framework", Microsoft Corporation
URL:http://msdn2.microsoft.com/en-us/library/aa679800(office.11).aspx

[9] "Office Information Bridge Framework 1.5 Solution Development Guide",
Microsoft Corporation
URL: http://www.microsoft.com/technet/solutionaccelerators/cits
/iwp/ibf/soldev/moibf1_sdg_01.mspx

[10] Ricard Roma i Dalfo "Information Bridge Framework: Bringing SOA to the Desktop in Office Applications", Microsoft Corporation
URL: http://msdn2.microsoft.com/en-us/library/aa480044.aspx

[11] Bryan Castle "Introduction to Web Services for Remote Portlets"
URL: http://www-128.ibm.com/developerworks/library/ws-wsrp/

[12] "Web Services for Remote Portlets Specification", OASIS 2003
URL: http://www.oasis-open.org/committees/download.php/3343
/oasis-200304-wsrp-specification-1.0.pdf

[13] "WSRP Whitepaper", OASIS 2003
URL: http://www.oasis-open.org/apps/org/workgroup/wsrp
/download.php/2634/WSRP

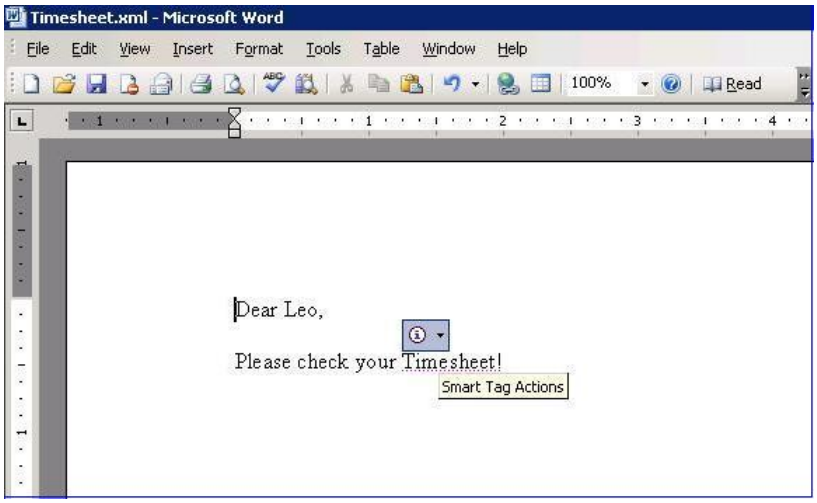
[14] Vijay Desai, Persistent Systems Private Limited
"Building a Simple IBF Solution - For Beginners"
URL:http://www.persistentsys.com/presentation
/InformationBridgeFramework.pdf

Appendix A

The screenshots of the testing

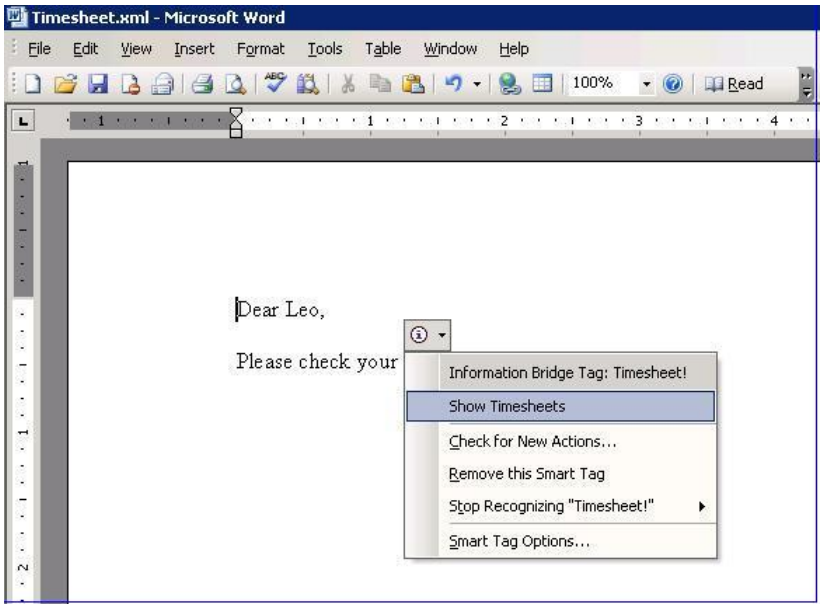
A.1 A smart tag icon

When the user places the cursor over the text “Timesheet”, a smart tag icon appears.



A.2 The smart tag menu

After clicking the smart tag icon, a menu opens.

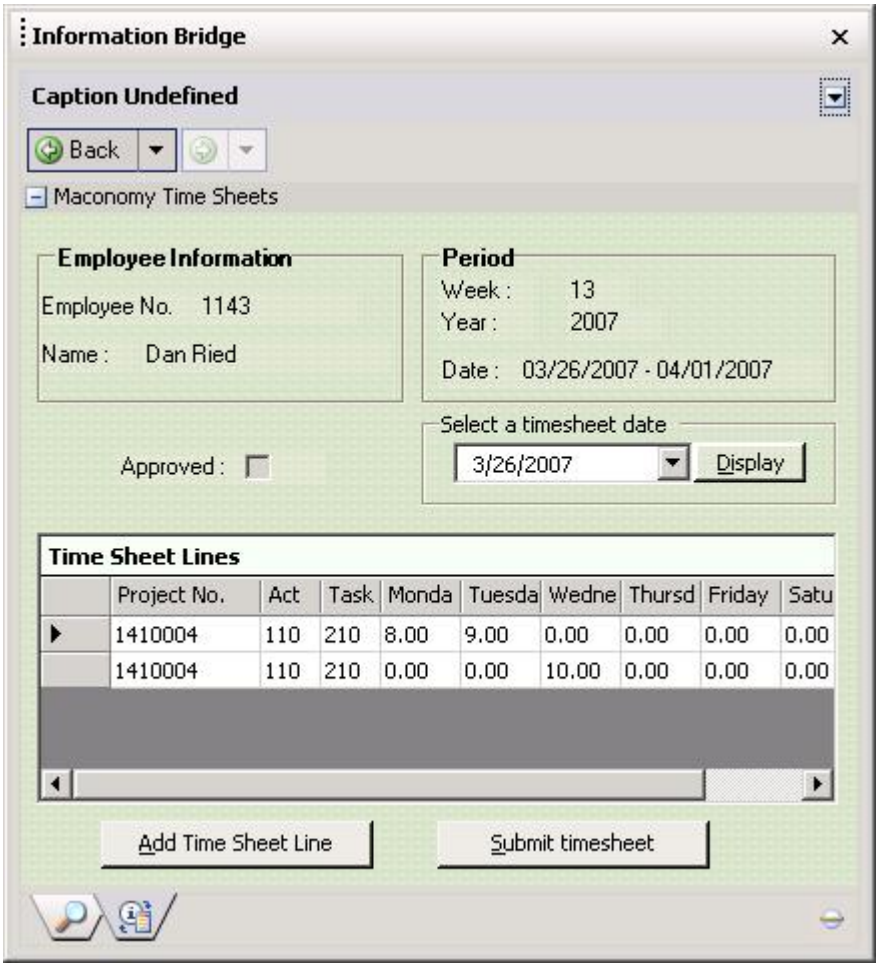


A.3 The “LogOn” region

When a user clicks the “Show Timesheets” menu item, the Office task pane titled “Maconomy Timesheet” appears containing a region that exposes “Maconomy Access Control” interface to the user. Once the user inputs the username and password, the “OK” button is enabled to let the user click it and then access Maconomy system.

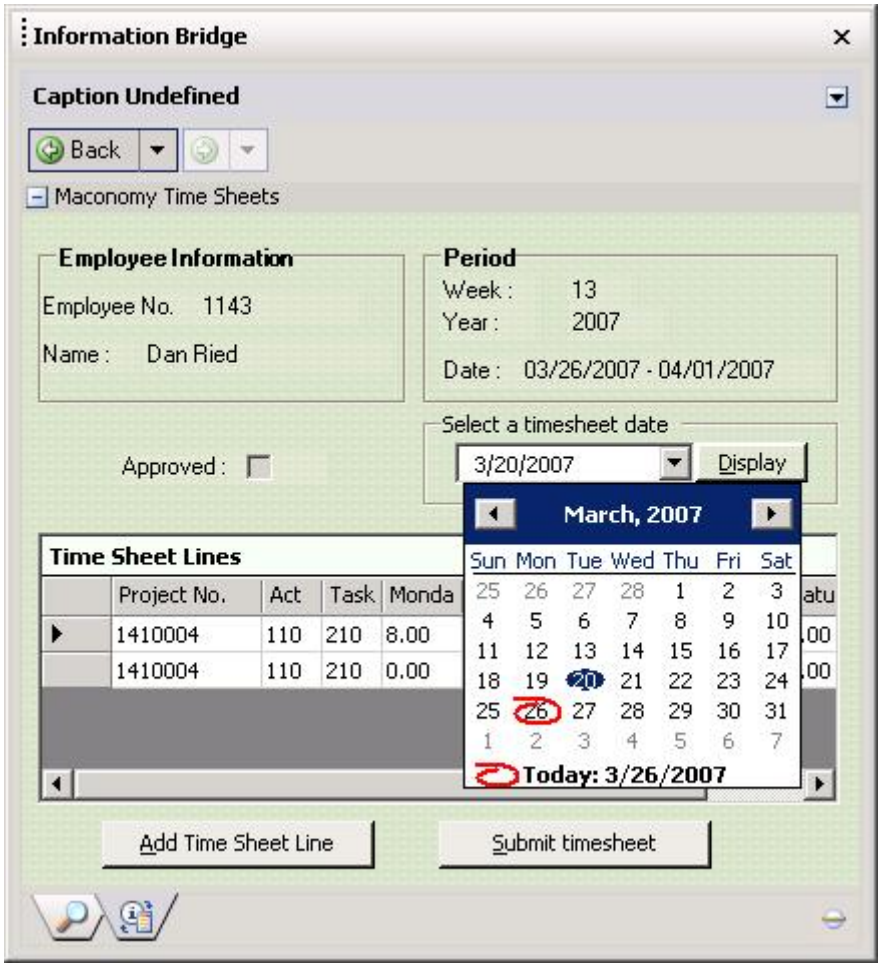
A.4 The "DisplayTimesheet" region

If the provided username and password are correct, the user's timesheet for the current week is displayed in a new Region. The "current" date is 3/26/2007 as shown in the screendump.

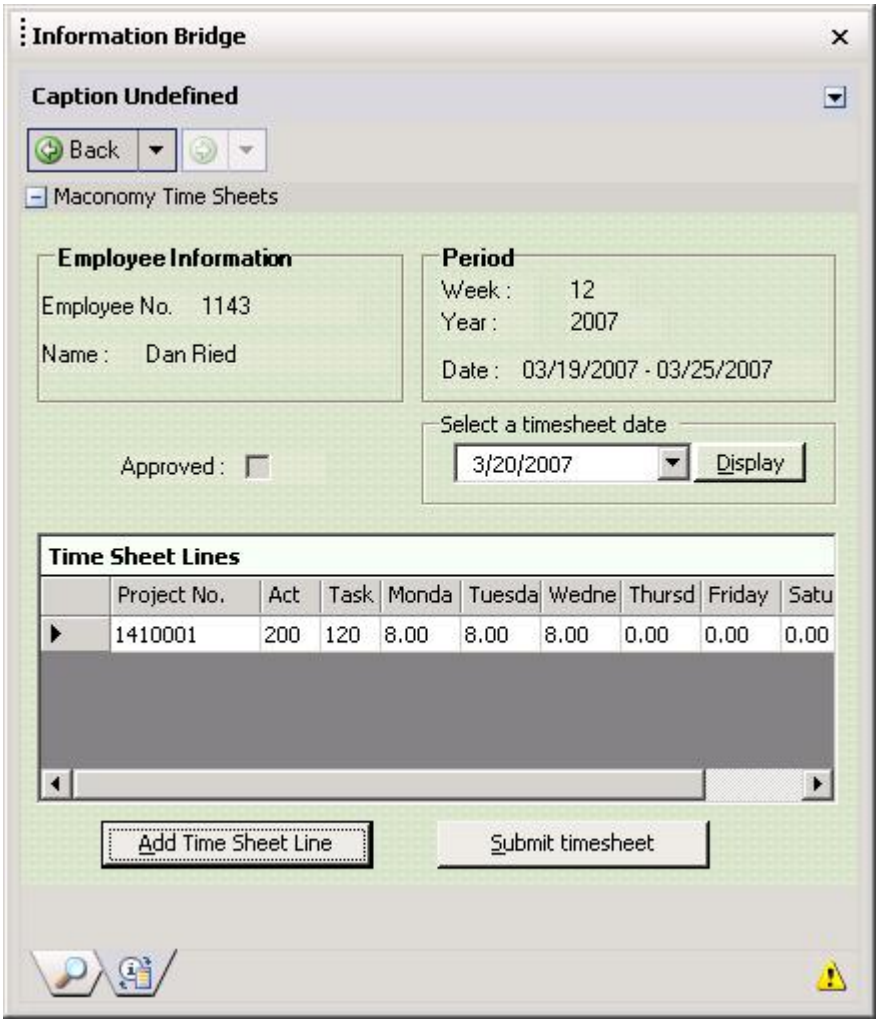


A.5 Viewing another timesheet

Now the user can view the timesheet data or switch to another timesheet by selecting a date and clicking the button “Display”. The new date selected is 3/20/2007 as shown in the screendump.

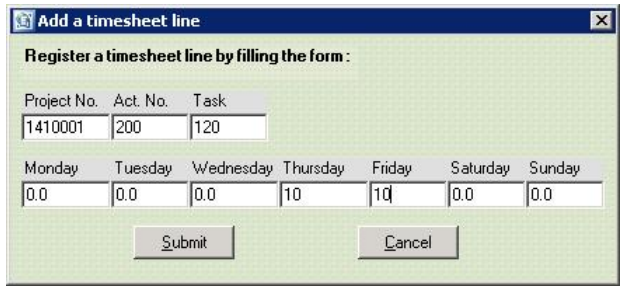


Then the timesheet for the period 03/19/2007 - 03/25/2007 is displayed.

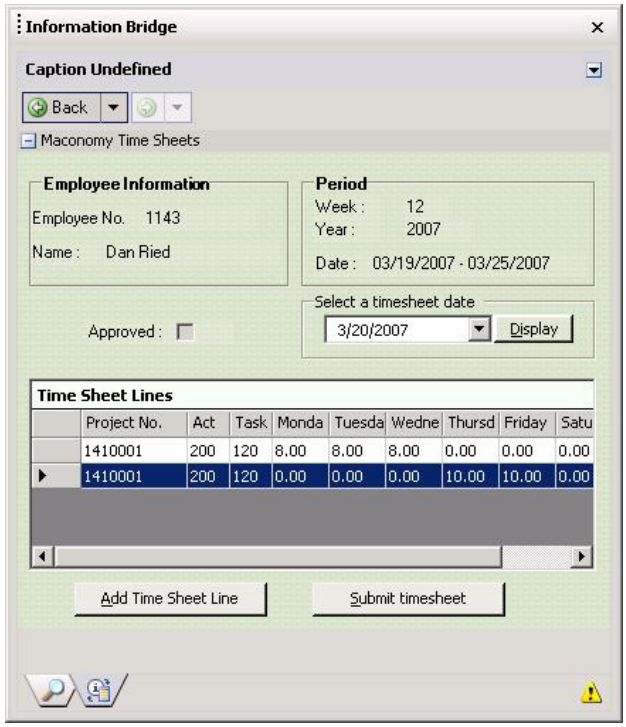


A.6 Adding a timesheet line

When the user clicks the button "Add Timesheet Line", a modal dialog box pops up to ask the user to input project no., task name, etc.



By clicking the button "Submit" in the modal dialog box, the dialog box is closed and a new timesheet line is added into the current timesheet. (The line in blue is the one added into the timesheet.)



Appendix B

A SmartTag

The following content - Timesheet.xml is the Word document mentioned in section 4.9. The lines in ***italics*** shown below were added for a smart tag, which is associated with the text "Timesheet".

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?mso-application progid="Word.Document"?>
<w:wordDocument
xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:w10="urn:schemas-microsoft-com:office:word"
xmlns:sl="http://schemas.microsoft.com/schemaLibrary/2003/core"
xmlns:aml="http://schemas.microsoft.com/aml/2001/core"
xmlns:wx="http://schemas.microsoft.com/office/word/2003/auxHint"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
xmlns:ns0="http://schemas.microsoft.com/InformationBridge/2004"
xmlns:st1="http://schemas.microsoft.com/InformationBridge/2004"
  w:macrosPresent="no"w:embeddedObjPresent="no"w:ocxPresent="no"
xml:space="preserve">
<o:SmartTagType
o:namespaceuri=
"http://schemas.microsoft.com/InformationBridge/2004"
```

```
o:url="http://InformationBridge/CRM/SmartTag/"
o:name="reference"/>
<o:DocumentProperties>
    <o:Title>Timesheet</o:Title>
    <o:Author>Liu Xuesong</o:Author>
    <o:LastAuthor>Liu Xuesong</o:LastAuthor>
    <o:Created>2007-03-17T13:22:00Z</o:Created>
    <o:LastSaved>2007-03-17T16:45:00Z</o:LastSaved>
    <o:Pages>1</o:Pages><o:Words>6</o:Words>
    <o:Characters>39</o:Characters>
    <o:Company>Maconomy A/S</o:Company>
    <o:Lines>1</o:Lines>
    <o:Paragraphs>1</o:Paragraphs>
    <o:CharactersWithSpaces>44</o:CharactersWithSpaces>
</o:DocumentProperties>
<w:fonts>
    <w:defaultFonts
    w:ascii="Times New Roman"
    w:fareast="Times New Roman"
    w:h-ansi="Times New Roman"
    w:cs="Times New Roman"/>
</w:fonts>
<w:styles>
    <w:versionOfBuiltInStylenames w:val="4"/>
    <w:latentStyles w:defLockedState="off"
                    w:latentStyleCount="156"/>
    <w:style w:type="paragraph" w:default="on"
     w:styleId="Normal">
        <w:name w:val="Normal"/>
        <w:rsid w:val="00933555"/>
        <w:rPr>
            <wx:font wx:val="Times New Roman"/>
            <w:sz w:val="24"/>
            <w:sz-cs w:val="24"/>
            <w:lang w:val="EN-US" w:fareast="EN-US"
             w:bidi="AR-SA"/>
        </w:rPr>
    </w:style>
    <w:style w:type="character" w:default="on"
             w:styleId="DefaultParagraphFont">
        <w:name w:val="Default Paragraph Font"/>
        <w:semiHidden/>
    </w:style>
    <w:style w:type="table" w:default="on"
```

```
    w:styleId="TableNormal">
      <w:name w:val="Normal Table"/>
      <wx:uiName wx:val="Table Normal"/>
      <w:semiHidden/>
      <w:rPr><wx:font wx:val="Times New Roman"/>
      </w:rPr>
      <w:tblPr>
        <w:tblInd w:w="0" w:type="dxa"/>
        <w:tblCellMar>
          <w:top w:w="0" w:type="dxa"/>
          <w:left w:w="108" w:type="dxa"/>
          <w:bottom w:w="0" w:type="dxa"/>
          <w:right w:w="108" w:type="dxa"/>
        </w:tblCellMar>
      </w:tblPr>
  </w:style>
  <w:style w:type="list" w:default="on" w:styleId="NoList">
      <w:name w:val="No List"/><w:semiHidden/>
  </w:style>
</w:styles>
<w:docPr>
  <w:view w:val="print"/>
  <w:zoom w:percent="100"/>
  <w:doNotEmbedSystemFonts/>
  <w:proofState w:spelling="clean"/>
  <w:attachedTemplate w:val=""/>
  <w:defaultTabStop w:val="720"/>
  <w:punctuationKerning/>
  <w:characterSpacingControl w:val="DontCompress"/>
  <w:optimizeForBrowser/><w:validateAgainstSchema/>
  <w:saveInvalidXML w:val="off"/>
  <w:ignoreMixedContent w:val="off"/>
  <w:alwaysShowPlaceholderText w:val="off"/>
  <w:compat>
      <w:breakWrappedTables/>
      <w:snapToGridInCell/>
      <w:wrapTextWithPunct/>
      <w:useAsianBreakRules/>
      <w:dontGrowAutofit/>
  </w:compat>
</w:docPr>
<w:body>
  <wx:sect>
      <w:p/>
```

```
        <w:p><w:r><w:t>Dear Nikolaj,</w:t></w:r></w:p>
        <w:p/>
        <w:p><w:r><w:t>Please check your </w:t></w:r>
       <ns0:reference data="&lt;?xml
       version=&quot;1.0&quot;
       encoding=&quot;utf-16&quot;?&gt;&#xA;&lt;
       ContextInformation
       xmlns:xsd=&quot;http://www.w3.org/2001/XMLSchema&quot;
       xmlns:
       xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
       MetadataScopeName=&quot;Maconomy.SampleSolution.TS&quot;
       EntityName=&quot;TimeSheet&quot;
       ViewName=&quot;LogonMaconomy&quot;
       ReferenceSchemaName=&quot;AccessSchema&quot;
        xmlns=&quot;
       http://schemas.microsoft.com/InformationBridge/2004/
       ContextInformation&quot;
       &gt;&#xA; &lt;Reference&gt;&#xA;  &
       lt;Dialog msibf:MetadataScopeName=&quot;
       Maconomy.SampleSolution.TS&quot;
       msibf:EntityName=&quot;TimeSheet&quot; Name=&quot;
       Timesheet&quot; xmlns=&quot;maconomy.com&quot;
       msibf:ViewName=&quot;LogonMaconomy&quot;
       msibf:ReferenceSchemaName=&quot;AccessSchema&quot;
       xmlns:msibf=&quot;
       http://schemas.microsoft.com/InformationBridge/2004&quot;
       /&gt;&#xA; &lt;/Reference&gt;
       &#xA;&lt;/ContextInformation&gt;" w:st="on">
            <w:r>
                <w:t>Timesheet!</w:t>
            </w:r>
         </ns0:reference>
        </w:p>
        <w:sectPr>
            <w:pgSz w:w="12240" w:h="15840"/>
            <w:pgMar w:top="1440" w:right="1800"
                w:bottom="1440"w:left="1800"w:header="720"
                w:footer="720"w:gutter="0"/>
            <w:cols w:space="720"/>
            <w:docGrid w:line-pitch="360"/>
        </w:sectPr>
    </wx:sect>
</w:body>
</w:wordDocument>
```

Appendix C

Source code of Maconomy Web service

This section contains the source code in M-Script used to define the data types and methods for the Maconomy Web service - "Timesheet".

```
#version 13

/*
* Copyright 2007 by Xuesong Liu
*
* Web service prototype with a collection of
   operations on the Maconomy
* timesheet dialog interface.
*/
package timesheet(1);


/*=======================================
  External web services operations
=======================================*/

  // We need to specify the type of a timesheet description
```

```
  //   a couple of times, so instead of rewriting it
  //   everytime, we just refer to the type below.
var timesheetType =
{
  type: "object",
  elements:
  {
    EmployeeNumber: "string",
    PeriodStart: "date",
    Approved: "bool",
    Rows:
    {
      type: "array",
      elements:
      {
        type: "object",
        elements:
        {
          JobNumber: "string",
          ActivityNumber: "string",
      TaskName: "string",
          NumberOfDay1: "real",
          NumberOfDay2: "real",
          NumberOfDay3: "real",
          NumberOfDay4: "real",
          NumberOfDay5: "real",
          NumberOfDay6: "real",
          NumberOfDay7: "real"
        }
      }
    }
  }
};

var referenceType =
{
  type: "object",
  elements:
  {
    username: "string",
    password: "string",
    PeriodStart: "date"
  }
};
```

```
var tsLineType =
{
  type: "object",
  elements:
  {
    JobNumber:      "string",
    ActivityNumber: "string",
    TaskName:       "string",
    NumberOfDay1:   "real",
    NumberOfDay2:   "real",
    NumberOfDay3:   "real",
    NumberOfDay4:   "real",
    NumberOfDay5:   "real",
    NumberOfDay6:   "real",
    NumberOfDay7:   "real"
  }
};

/*-------------------------------------------------------
  General service description.
  Here we specify various meta data about the web service
  as a whole.
-------------------------------------------------------*/

public var serviceDescription =
{
   // The service namespace defines which namespace to put
   // the service data into
   // Used among other things as the "target namespace" for
   // the WSDL generator
  serviceNamespace: "maconomy.com",

   //A short documentation text inserted as a "<documentation>"
   //tag in the service part of the WSDL
  documentation: "A collection of operations for "+
                 "Maconomy Timesheet data handling."
};

/*-----------------------------------------------------
  Operation:  get

  Input:     object of type "referenceType".
  Output:    object of type "timesheetType".
```

```
-------------------------------------------------------*/
// The meta data for the "get" function.  This must be a public
// object named as "<functionName>Info" in order for the web
// service framework to locate it.

public var getInfo =
{
  // This operation requires a session ID in the SOAP header
  requiresSession: false,

  // Here we specify the data type of each of the parameters to
  // the function.The "inputTypes" object contains one property
  // for each of the parameters
  // of the function.
  inputTypes:
  {
    refer: referenceType
  },

  // Here we specify the output type of the function.  In the
  // case it is the timesheet data.
  outputType: timesheetType,

  // Documentation to be inserted as a "<documentation>" tag
  // in the operation description in the WSDL.
  documentation: 'DOC'
This operation opens a timesheet dialog and get timesheet
data for a specific timesheet.
DOC
};

// Here at last we get to write our web service function.
public function get(refer)
{
newsession();
var result;
var loginResult=maconomy::login(refer.username,refer.password);

// To get the employee number of the current user
var sqlResult = maconomy::sql('SQL', 1, null, refer.username);
select employeenumber from employee where name1='^1'
SQL
var res = sqlResult.result.rows[0].employeenumber.value;
```

```
if (loginResult.ok)
  {
    var id = maconomy::dialogOpen("TimeSheets");
    var key = {
     EmployeeNumber: {value : res },
     PeriodStart: {value : refer.PeriodStart}
    };

    result = timesheetFilter( maconomy::dialogGet(id, key) );
    maconomy::dialogClose(id);
    maconomy::logout();
  }

deletesession();
return result;
}

/*-------------------------------------------------------------
Operation:  register

Input:  a timesheet registration on any timesheet as specified
Output:  true.
-------------------------------------------------------------*/

public var registerInfo =
{
  requiresSession: false,

  inputTypes:
  {
    refer: referenceType,
    entry: tsLineType
  },

  outputType: "bool",

  documentation: 'DOC'
This operation adds a new timesheet registration on an
existing timesheet. It is not necessary to call the get()
operation first since register()manages opening and closing
of the timesheet dialog itself.
DOC
};
```

```
public function register(refer,entry)
{
newsession();
var loginResult=maconomy::login(refer.username,refer.password);

// To get the employee number of the current user
var sqlResult = maconomy::sql('SQL', 1, null, refer.username);
select employeenumber from employee where name1='^1'
SQL
var res = sqlResult.result.rows[0].employeenumber.value;

if (loginResult.ok)
{
    var id = maconomy::dialogOpen("TimeSheets");
    var key = {
     EmployeeNumber: {value : res },
     PeriodStart: {value : refer.PeriodStart}
    };

    try
    {
      maconomy::dialogGet(id, key);

      maconomy::dialogNewLower(id);

      maconomy::dialogPutLower(id,
       {
         JobNumber:      { value: entry.JobNumber },
         ActivityNumber: { value: entry.ActivityNumber },
         TaskName:       { value: entry.TaskName },
         NumberDay1:   { value: entry.NumberOfDay1 },
         NumberDay2:   { value: entry.NumberOfDay2 },
         NumberDay3:   { value: entry.NumberOfDay3 },
         NumberDay4:   { value: entry.NumberOfDay4 },
         NumberDay5:   { value: entry.NumberOfDay5 },
         NumberDay6:   { value: entry.NumberOfDay6 },
         NumberDay7:   { value: entry.NumberOfDay7 }
       });

      maconomy::commit();
    }
    catch (e)
    {
      var f = file::open("exceptionDebug.txt", "a");
```

```
      file::dumpvalue(f, e);
      file::close(f);
    }

    maconomy::dialogClose(id);
    maconomy::logout();
    deletesession();
    return true;
}
else
{
   deletesession();
   return false;
}
}

/*==============================
  Internal helpers
================================*/

// Convert all properties on an object to { value:  xxx } format
function addValues(o)
{
  var result = {};
  for (var name in o)
    new result[name] = { value: o[name] };
  return result;
}


  // Select specific fields from dialog data
function timesheetFilter(dialogResult)
{
  var dialogData  = dialogResult.dialogData;
  var upperRecord = dialogData.upperPane.rows[0];
  var lowerRows   = dialogData.lowerPane.rows;

    // Copy individual lower pane fields into return value
  var newLowerRows = new [sizeof(lowerRows)];
  for (var i in lowerRows)
  {
    var lowerRecord = lowerRows[i];
    newLowerRows[i] =
    {
```

```
      JobNumber:       lowerRecord.JobNumber.value,
      ActivityNumber: lowerRecord.ActivityNumber.value,
      TaskName:        lowerRecord.TaskName.value,
      NumberOfDay1:    lowerRecord.NumberDay1.value,
      NumberOfDay2:    lowerRecord.NumberDay2.value,
      NumberOfDay3:    lowerRecord.NumberDay3.value,
      NumberOfDay4:    lowerRecord.NumberDay4.value,
      NumberOfDay5:    lowerRecord.NumberDay5.value,
      NumberOfDay6:    lowerRecord.NumberDay6.value,
      NumberOfDay7:    lowerRecord.NumberDay7.value
    };
  }

  // Copy individual upper pane fields into return value
  var newUpperRecord =
  {
    EmployeeNumber: upperRecord.EmployeeNumber.value,
    PeriodStart:    upperRecord.PeriodStart.value,
    Approved:       upperRecord.Approved.value
  };

  return newUpperRecord + { Rows: newLowerRows };
}
```

Appendix D

Metadata for the prototype solution

The following XML file contains the definition of the IBF metadata defined for the prototype solution.

MSIBFMetadata.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Metadata Version="1.0.0.0"
xmlns="http://schemas.microsoft.com/InformationBridge/2004/M
etadata">
  <MetadataScopes
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <MetadataScope Name="InformationBridge"
Version="1.0.0.0"DefaultEntityName="Search">
      <Entities>
        <Entity Name="GenericSmartTags"
DefaultViewName="GenericRecognizers">
          <Views>
            <View Name="GenericActionHandlers"
SchemaName="MSIBF.GenericActionHandler"IsList="false">
```

```
                <MenuItemDefinition MenuType="0">
                </MenuItemDefinition>
                <Actions>
                  <Action Name="YourActionName"
Type="EnterContext"
InputSchemaName="MSIBF.GenericActionHandler"
IsSequentialExecution="true">
                    <MenuItemDefinition MenuType="0">
                    </MenuItemDefinition>
                    <OperationInstances>
                      <OperationInstance
OperationName="LoadActionHandler"Index="1">
                        <TransformationInstances>
                          <TransformationInstance
TransformationName="GenericActionHandler"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                          </TransformationInstance>
                        </TransformationInstances>
                      </OperationInstance>
                    </OperationInstances>
                  </Action>
                </Actions>
              </View>
              <View Name="GenericRecognizers"
SchemaName="MSIBF.GenericRecognizer"IsList="false">
                <MenuItemDefinition MenuType="0">
                </MenuItemDefinition>
                <Actions>
                  <Action Name="YourActionName"
Type="EnterContext"
InputSchemaName="MSIBF.GenericRecognizer"
IsSequentialExecution="true">
                    <MenuItemDefinition MenuType="0">
                    </MenuItemDefinition>
                    <OperationInstances>
                      <OperationInstance
OperationName="LoadRecognizer"Index="1">
                        <TransformationInstances>
                          <TransformationInstance
TransformationName="GenericRecognizer"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                          </TransformationInstance>
                        </TransformationInstances>
                      </OperationInstance>
```

```
                    <OperationInstance
OperationName="ActivateRecognizer"Index="2">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="GenericRecognizer"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
              </Actions>
            </View>
          </Views>
        </Entity>
        <Entity Name="Search"
DefaultReferenceSchemaName="NullReference"
DefaultViewName="Search">
          <Views>
            <View Name="Search" SchemaName="MSIBF.Search"
IsList="false">
              <MenuItemDefinition MenuType="0">
              </MenuItemDefinition>
              <ViewLocators>
                <ViewLocator
ReferenceSchemaName="NullReference">
                  <OperationInstance
OperationName="LoadSearchFromSearchPort"Index="0">
                    <TransformationInstances>
                      <TransformationInstance
TransformationName="LoadSearchFromSearchPort"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                      </TransformationInstance>
                    </TransformationInstances>
                  </OperationInstance>
                </ViewLocator>
              </ViewLocators>
              <Actions>
                <Action
Name="ShowDefaultSearchCriteriaRegion"
IsSequentialExecution="false">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
```

```
                    <OperationInstance
OperationName="BuildDefaultSearchCriteriaRegion"Index="1">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="-1001"
InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                    <OperationInstance
OperationName="ShowRegion"Index="2">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="1"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
                <Action Name="ShowSearchCriteriaRegion"
InputSchemaName="MSIBF.SearchType"
IsSequentialExecution="false">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="QuerySearchTypes"Index="1">
                    </OperationInstance>
                    <OperationInstance
OperationName="BuildSearchSchema"Index="2">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="BuildSearchFromQueryMenu"Index="1"
InputOperationIndex="1"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                        <TransformationInstance Index="2"
InputOperationIndex="-1"
InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                    <OperationInstance
OperationName="BuildSearchCriteriaRegion"Index="3">
                      <TransformationInstances>
```

```
                        <TransformationInstance Index="1"
InputOperationIndex="2"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                        <TransformationInstance Index="2"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                    <OperationInstance
OperationName="ShowRegion"Index="4">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="3"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
                <Action Name="ShowSearchPickerRegion"
IsSequentialExecution="false">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="QuerySearchTypes"Index="1">
                    </OperationInstance>
                    <OperationInstance
OperationName="BuildSearchSchema"Index="2">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="BuildSearchFromQueryMenu"Index="1"
InputOperationIndex="1"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                        <TransformationInstance Index="2"
InputOperationIndex="-1"
InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                    <OperationInstance
OperationName="BuildSearchPickerRegion"Index="3">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="2"InputOperationOutputSchemaIndex="1">
```

```
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                    <OperationInstance
OperationName="ShowRegion"Index="4">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="3"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
                <Action Name="ShowSearchResultsRegion"
InputSchemaName="MSIBF.SearchType"
IsSequentialExecution="false">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="ShowRegion"Index="1">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="BuildSearchResultsRegion"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
              </Actions>
            </View>
          </Views>
        </Entity>
      </Entities>
      <Ports>
        <Port xsi:type="PortFileXml"
Name="BuildDefaultSearchCriteriaRegionXslt"IsCached="false"
AuthenticationTypeValue="None">
          <Data>
            <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:translator="http://schemas.microsoft.com/InformationBr
idge/2004/Translator"
```

```
xmlns:inputs="http://schemas.microsoft.com/InformationBridge
/2004/Inputs"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <xsl:output method="xml" encoding="utf-8" />
                <xsl:template match="/">
                  <xsl:apply-templates
select="inputs:GetRootXPath(1)/msibf:SearchTypes"/>
                </xsl:template>
                <xsl:template match="msibf:SearchTypes">
                  <msibf:Region Enabled="true"
MetadataScopeName="InformationBridge"EntityName="Search">
                    <msibf:RegionProperties
Caption="{translator:Translate('SearchCriteriaRegionCaption'
)}"
Description="{translator:Translate('SearchCriteriaRegionDesc
ription')}"ShowAs="ExpandedRegion"
TypeName="Microsoft.InformationBridge.Framework.UI.Controls.
SearchCriteriaRegion">
                      <xsl:copy-of select="msibf:SearchType"
/>
                    </msibf:RegionProperties>
                  </msibf:Region>
                </xsl:template>
              </xsl:stylesheet>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="BuildSearchCriteriaRegionXslt"IsCached="false"
AuthenticationTypeValue="None">
            <Data>
              <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:translator="http://schemas.microsoft.com/InformationBr
idge/2004/Translator"
xmlns:inputs="http://schemas.microsoft.com/InformationBridge
/2004/Inputs"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <xsl:output method="xml" encoding="utf-8" />
                <xsl:template match="/">
                  <xsl:apply-templates
select="inputs:GetRootXPath(1)/msibf:SearchTypes"/>
                </xsl:template>
```

```
                <xsl:template match="msibf:SearchTypes">
                  <xsl:choose>
                    <xsl:when
test="count(msibf:SearchType)=1">
                      <xsl:apply-templates
select="msibf:SearchType"/>
                    </xsl:when>
                    <xsl:otherwise>
                      <xsl:apply-templates
select="msibf:SearchType[@Name=inputs:GetRootXPath(2)/@Name]
"/>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:template>
                <xsl:template match="msibf:SearchType">
                  <msibf:Action Enabled="true">
                    <xsl:apply-templates
select="msibf:SearchCriteria"/>
                    <msibf:Search>
                      <msibf:SearchTypes>
                        <xsl:copy-of select="." />
                      </msibf:SearchTypes>
                    </msibf:Search>
                    <msibf:RegionProperties
Caption="{translator:Translate('SearchCriteriaRegionCaption'
)}"
Description="{translator:Translate('SearchCriteriaRegionDesc
ription')}"ShowAs="ExpandedRegion"/>
                  </msibf:Action>
                </xsl:template>
                <xsl:template match="msibf:SearchCriteria">
                  <xsl:choose>
                    <xsl:when test="@ActionName">
                      <xsl:attribute name="AssociationName">
                        <xsl:value-of select="@ActionName" />
                      </xsl:attribute>
                      <xsl:apply-templates
select="@MetadataSolutionName"/>
                      <xsl:apply-templates
select="@MetadataScopeName"/>
                      <xsl:apply-templates
select="@EntityName"/>
                      <xsl:apply-templates select="@ViewName"
/>
```

```
                    <xsl:apply-templates
select="@ReferenceSchemaName"/>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:attribute
name="AssociationName">ShowDefaultSearchCriteriaRegion</xsl:
attribute>
                    <xsl:attribute
name="MetadataScopeName">InformationBridge</xsl:attribute>
                    <xsl:attribute
name="EntityName">Search</xsl:attribute>
                    <xsl:attribute
name="ViewName">Search</xsl:attribute>
                    <xsl:attribute
name="ReferenceSchemaName">SearchType</xsl:attribute>
                  </xsl:otherwise>
                </xsl:choose>
              </xsl:template>
              <xsl:template match="@*">
                <xsl:copy />
              </xsl:template>
            </xsl:stylesheet>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="BuildSearchFromQueryMenuXslt"IsCached="false"
AuthenticationTypeValue="None">
          <Data>
            <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
              <xsl:output method="xml" encoding="utf-8" />
              <xsl:template match="/msibf:Associations">
                <Search
xmlns="http://schemas.microsoft.com/InformationBridge/2004">
                  <SearchTypes>
                    <xsl:for-each select="msibf:Action">
                      <xsl:copy-of select="msibf:SearchType"
/>
                    </xsl:for-each>
                  </SearchTypes>
                </Search>
              </xsl:template>
```

```
            </xsl:stylesheet>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="BuildSearchPickerRegionXslt"IsCached="false"
AuthenticationTypeValue="None">
          <Data>
            <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:translator="http://schemas.microsoft.com/InformationBr
idge/2004/Translator"
xmlns:inputs="http://schemas.microsoft.com/InformationBridge
/2004/Inputs"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
              <xsl:output method="xml" encoding="utf-8" />
              <xsl:template match="/">
                <xsl:apply-templates
select="inputs:GetRootXPath(1)/msibf:SearchTypes"/>
              </xsl:template>
              <xsl:template match="msibf:SearchTypes">
                <xsl:choose>
                  <xsl:when
test="count(msibf:SearchType)=1">
                    <msibf:Action Enabled="true"
AssociationName="ShowSearchCriteriaRegion"
MetadataScopeName="InformationBridge"EntityName="Search"
ViewName="Search"ReferenceSchemaName="NullReference">
                      <msibf:RegionProperties
Caption="{translator:Translate('SearchCriteriaRegionCaption'
)}"
Description="{translator:Translate('SearchCriteriaRegionCapt
ionDescription')}"ShowAs="ExpandedRegion"/>
                    </msibf:Action>
                  </xsl:when>
                  <xsl:otherwise>
                    <msibf:Region Enabled="true"
MetadataScopeName="InformationBridge"EntityName="Search">
                      <msibf:RegionProperties
Caption="{translator:Translate('SearchPickerRegionCaption')}
"
Description="{translator:Translate('SearchPickerRegionDescri
ption')}"ShowAs="ExpandedRegion"
TypeName="Microsoft.InformationBridge.Framework.UI.Controls.
```

```
SearchPickerRegion">
                        <msibf:Search>
                          <xsl:for-each
select="msibf:SearchType">
                            <xsl:copy>
                              <xsl:for-each select="@*">
                                <xsl:copy />
                              </xsl:for-each>
                            </xsl:copy>
                          </xsl:for-each>
                        </msibf:Search>
                      </msibf:RegionProperties>
                    </msibf:Region>
                  </xsl:otherwise>
                </xsl:choose>
              </xsl:template>
            </xsl:stylesheet>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="BuildSearchResultsRegionXslt"IsCached="false"
AuthenticationTypeValue="None">
          <Data>
            <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:inputs="http://schemas.microsoft.com/InformationBridge
/2004/Inputs"
xmlns:translator="http://schemas.microsoft.com/InformationBr
idge/2004/Translator"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
              <xsl:output method="xml" encoding="utf-8" />
              <xsl:template match="/msibf:SearchType">
                <msibf:Action Enabled="true"
AssociationName="{@ActionName}">
                  <xsl:apply-templates select="@ActionType"
/>
                  <xsl:apply-templates
select="@MetadataSolutionName"/>
                  <xsl:apply-templates
select="@MetadataScopeName"/>
                  <xsl:apply-templates select="@EntityName"
/>
                  <xsl:apply-templates select="@ViewName" />
```

```
                    <xsl:apply-templates
select="@ReferenceSchemaName"/>
                    <xsl:copy-of
select="msibf:InputReferenceInstance/*"/>
                    <msibf:RegionProperties
Caption="{translator:Translate('SearchResultsRegionCaption')
}"
Description="{translator:Translate('SearchResultsRegionDescr
iption')}"ShowAs="ExpandedRegion"/>
                  </msibf:Action>
                </xsl:template>
                <xsl:template match="@*">
                  <xsl:copy />
                </xsl:template>
              </xsl:stylesheet>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="BuildSearchSchemaXslt"IsCached="false"
AuthenticationTypeValue="None">
            <Data>
              <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:translator="http://schemas.microsoft.com/InformationBr
idge/2004/Translator"
xmlns:inputs="http://schemas.microsoft.com/InformationBridge
/2004/Inputs"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <xsl:output method="xml" encoding="utf-8" />
                <xsl:template match="/">
                  <Search
xmlns="http://schemas.microsoft.com/InformationBridge/2004">
                    <SearchTypes>
                      <xsl:for-each
select="inputs:GetRootXPath(1)/msibf:SearchTypes/msibf:Searc
hType">
                        <xsl:copy-of select="." />
                      </xsl:for-each>
                      <xsl:for-each
select="inputs:GetRootXPath(2)/msibf:SearchTypes/msibf:Searc
hType">
                        <xsl:copy-of select="." />
                      </xsl:for-each>
```

```
                    </SearchTypes>
                  </Search>
                </xsl:template>
              </xsl:stylesheet>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="GenericActionHandler"IsCached="false"
AuthenticationTypeValue="Windows">
            <Data>
              <msibf:GenericActionHandler
TypeName="Maconomy.Sample.SmartTag.TimeSheetActionHandler"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <YourInitializationData />
              </msibf:GenericActionHandler>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="GenericRecognizer"IsCached="false"
AuthenticationTypeValue="Windows">
            <Data>
              <msibf:GenericRecognizer Id="YourRecognizer"
TypeName="YourNamespace.YourRecognizerTypeName"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <YourInitializationData />
              </msibf:GenericRecognizer>
            </Data>
          </Port>
          <Port xsi:type="PortAssembly"
Name="GenericSmartTags"
Location="D:\Thesis\Sample\SmartTag\bin\Debug\SmartTag.dll"
IsCached="true"AuthenticationTypeValue="None">
          </Port>
          <Port xsi:type="PortFileXml" Name="MSIBF.Schemas"
IsCached="false"AuthenticationTypeValue="Windows">
            <Data>
              <xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified"
targetNamespace="http://schemas.microsoft.com/InformationBri
dge/2004"xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
```

```
<!--
Root Element:   Associations
Operation:      QueryMenu, Custom/ShowAssociations.
Commands:       QueryMenu, Navigation.
Description:    The QueryMenu schema is UI agnostic;
the ShowAssociations schema is its extension.  It
is easier to read and write the shared schema then to use
extensions or substitution groups, but some parts of it
might not be ever   used/generated by the QueryMenu operation.
-->
              <xs:element name="Associations">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Properties">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element
ref="msibf:ViewProperties"minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:choice minOccurs="0"
maxOccurs="unbounded">
                      <xs:element ref="msibf:Action" />
                      <xs:element ref="msibf:Region" />
                      <xs:element ref="msibf:Relationship"
/>
                      <xs:element ref="msibf:ViewReference"
/>
                    </xs:choice>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   Region
        Operation:
Custom/ShowRegion, Custom/ShowDialog
        Descripyion:    The region can appear
        as a part of the associations sequence or
standalone.
    -->
              <xs:element name="Region"
type="msibf:AssociationType"/>
              <xs:element name="Relationship"
```

```
type="msibf:AssociationType"/>
              <xs:element name="ViewReference"
type="msibf:AssociationType"/>
              <xs:element name="Action">
                <xs:complexType>
                  <xs:complexContent mixed="true">
                    <xs:extension
base="msibf:AssociationType">
                      <xs:attribute name="ActionType" />
                    </xs:extension>
                  </xs:complexContent>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   Properties
        Operation:  ShowAssociations
        Descripyion:    Description for the pane
that shows associations
    -->
              <xs:element name="ViewProperties"
type="msibf:ViewPropertiesType"/>
              <!--
        Root Element:   Extension
        Operation:
Custom/LoadExtension
    -->
              <xs:element name="Extension">
                <xs:complexType>
                  <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   Text
        Operation:
Custom/InsertText
    -->
              <xs:element name="Text" type="xs:string" />
              <!--
        Root Element:   Message
        Operation:
Custom/ShowMessage
    -->
              <xs:element name="Message">
```

```
                <xs:complexType>
                  <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   AnnotatedReferences

Operation:      Custom/InsertResults
        Description:
Annotated references wraps multiple AnnotatedReference
elements
    -->
              <xs:element name="AnnotatedReferences">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element
ref="msibf:AnnotatedReference"minOccurs="0"
maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   AnnotatedReference

Operation:      Custom/InsertReference, Custom/ShowContext

Description:    Annotated reference wraps the reference
itself, together with caption
                        attributes and
optional text.  The name of the reference element is not

known, but its position does not ever change.
    -->
              <xs:element name="AnnotatedReference">
                <xs:complexType>
                  <xs:sequence>
<!-- Substitution is required for this
to validate the abstract reference -->
                    <xs:element
ref="msibf:AbstractReference"/>
                    <xs:element ref="msibf:Text"
minOccurs="0"/>
                  </xs:sequence>
```

```
                    <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
                  </xs:complexType>
                </xs:element>
                <!--
          Root Element:   GenericRecognizer

Operation:      Custom/LoadRecognizer,
Custom/ActivateRecognizer, Custom/DeactivateReconizer
Description:    Describes a generic recognizer.
When calling into the Activate/Deactivate
                          methods, the ID
attribute is the only one used - the other nodes are
ignored.
                          The arbitrary element is passed to the
Initialize method of the IRecognizer interface.
    -->
                <xs:element name="GenericRecognizer">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:any minOccurs="0" maxOccurs="1" />
                    </xs:sequence>
                    <xs:attribute default="" name="Id"
type="xs:string"/>
                    <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                  </xs:complexType>
                </xs:element>
                <!--
          Root Element:   GenericActionHandler

Operation:      Custom/LoadActionHandler
Description:    Describes a generic action handler.
The arbitrary element is passed to the Initialize
method of the IActionHandler interface.
    -->
                <xs:element name="GenericActionHandler">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:any minOccurs="0" maxOccurs="1" />
                    </xs:sequence>
                    <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                  </xs:complexType>
```

```
            </xs:element>
   <!--
        Operation:      Custom/GetProperties
Description:    Describes the properties requested and
returned from the MSIBF.UI.GetProperties custom operation
type.   -->
            <xs:element name="Properties">
              <xs:complexType>
                <xs:sequence minOccurs="0"
maxOccurs="unbounded">
                  <xs:element name="Property">
                    <xs:complexType>
                      <xs:attribute name="Name"
type="xs:string"/>
                      <xs:attribute name="Value"
type="xs:string"/>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <!--
        Element:        AbstractReference

Description:    The global element defining the
substitution group.  Because it is abstract,
it cannot appear in the XML and it needs substitution.
    -->
            <xs:element name="AbstractReference"
type="msibf:AbstractReferenceType"/>
            <!--
        Complex Type:   ViewPropertiesType
Description:    UI hints for the pane that
hosts associations.  The ShowAssociations operation
assumes (but does not require) the presence of
the UI hints, but otherwise the schema of the
content is open.
    -->
            <xs:complexType name="ViewPropertiesType">
              <xs:sequence>
                <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
              </xs:sequence>
              <xs:attribute name="Caption"
```

```
type="xs:string"use="optional"default=""/>
                <xs:attribute name="Description"
type="xs:string"use="optional"default=""/>
              </xs:complexType>
              <!--
        Complex Type:   AssociationType

Description:    Each association must have the
attributes as defined by the QueryMenu operation.
The ShowAssociations operation assumes
(but does not require)the presence of
the UI hints, but otherwise the
schema of the content is open.
    -->
              <xs:complexType name="AssociationType"
mixed="true">
                <xs:sequence>
<!-- The data to be passed to the region
may contain andmust contain the reference.
Substitution is required for this to
validate the abstract reference.
-->
                  <xs:element ref="msibf:AbstractReference"
/>
                  <!-- The data to be passed to the region
may contain properties of type RegionPropertiesType -->
                  <xs:element ref="msibf:RegionProperties"
minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup
ref="msibf:AssociationAttributes"/>
              </xs:complexType>
              <!--
        Complex Type:   AbstractReferenceType
Description:    A view reference type defined as open,
so that it can contain any elements and attributes.
This might not be suitable in general,
so it cannot be used as the base type for references.
Since this type is abstract it cannot be istantiated
but a XML element instance can be substituted wit
h this type.  An example of a reference and it's substitution
:   <xs:element name="myReference"
substitutionGroup="msibf:AbstractReference"
type="msibf:AbstractReferenceType" />
```

```
<myReference msibf:MetadataScopeName="..." ...  />
-->
              <xs:complexType name="AbstractReferenceType"
mixed="true">
                <xs:complexContent>
                  <xs:extension
base="msibf:AbstractReferenceAttributesType">
                    <xs:sequence>
                      <xs:any minOccurs="0"
maxOccurs="unbounded"namespace="##any"
processContents="lax"/>
                    </xs:sequence>
                    <xs:anyAttribute namespace="##any"
processContents="lax"/>
                  </xs:extension>
                </xs:complexContent>
              </xs:complexType>
              <!--
        Complex Type:   RegionPropertiesType
Description:    Some region properties are predefined,
but additional attributes might be used.
    -->
              <xs:complexType name="RegionPropertiesType">
                <xs:sequence>
                  <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup
ref="msibf:RegionAttributes"/>
                <xs:anyAttribute namespace="##any"
processContents="lax"/>
              </xs:complexType>
              <xs:element name="RegionProperties"
type="msibf:RegionPropertiesType"/>
              <!--
        Complex Type:
AbstractReferenceAttributesType
        Description:    It is not
possible in XSD to define the attributes for an element
without specifying the element name.  This is a
problem for view references - the solution is to
use substitution groups.  The abstract
reference type defines the common subtype -
all the elements in the substitution group
```

```
must use a same type or a derived type.
The elements in thesubstitution group
could use any element name.
    -->
                <xs:complexType
name="AbstractReferenceAttributesType"abstract="true">
                  <xs:attribute name="MetadataScopeName"
form="qualified"type="xs:string"use="required"/>
                  <xs:attribute name="EntityName"
form="qualified"type="xs:string"use="optional"/>
                  <xs:attribute name="ViewName"
form="qualified"type="xs:string"use="optional"/>
                  <xs:attribute name="ReferenceSchemaName"
form="qualified"type="xs:string"use="optional"/>
                </xs:complexType>
<!--   Element:        Credentials
Description:    This is the schema of the
authentication data that the
AuthenticationDefinitionOperationInstance
of a Port must comply with.
    -->
                <xs:element name="Credentials">
                  <xs:complexType>
                    <xs:all>
                      <xs:element name="UserName"
type="xs:string"/>
                      <xs:element name="Password"
type="xs:string"/>
                      <xs:element name="DomainName"
type="xs:string"minOccurs="0"/>
                    </xs:all>
                  </xs:complexType>
                </xs:element>
                <!--
        Element:        Search
        Description:
This is the schema of the metadata for Search.

It contains a list of search types.
    -->
                <xs:element name="Search">
                  <xs:complexType>
                    <xs:sequence>
<!-- This is the list of search types
```

```
(actions) available for the solution -->
                    <xs:element name="SearchTypes">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element ref="msibf:SearchType"
minOccurs="0"maxOccurs="unbounded"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <!--
        Element:        SearchType
Description:    This is the schema of the one search type.
A search type is used to define one individual search
action, the associated reference used to invoke
the action, and the search criteria UI used to
collect user input.
    -->
              <xs:element name="SearchType">
                <xs:complexType>
                  <xs:sequence>
                    <!--
                  This is the xml framgent
of the search criteria.  It gets filed with user data
and is used as reference for the search action invocation.
-->
                    <xs:element
name="InputReferenceInstance"minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="Namespaces"
use="required"/>
                      </xs:complexType>
                    </xs:element>
                    <!--
This is the list of search
criteria UI elements that persist data in the
reference instance defined above.
                --->
```

```
                    <xs:element name="SearchCriteria"
minOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
  <!-- This is the definition of one
input control -->
                          <xs:element name="SearchCriterion"
minOccurs="0"maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <!-- This is the list of
possible values that
                                     pre-populate the
list input control -->
                                <xs:element name="Item"
minOccurs="0"maxOccurs="unbounded">
                                  <xs:complexType>
                                    <xs:simpleContent>
                                      <xs:extension
base="xs:string">
                                        <xs:attribute
name="Value"type="xs:string"use="optional"/>
                                        <xs:attribute
name="Selected"type="xs:boolean"use="optional"/>
                                      </xs:extension>
                                    </xs:simpleContent>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                              <xs:attribute name="Caption"
type="xs:string"use="required"/>
                              <xs:attribute
name="Description"type="xs:string"use="required"/>
                              <xs:attribute name="Type"
use="required">
                                <xs:simpleType>
                                  <xs:restriction
base="xs:string">
                                    <xs:enumeration
value="text"/>
                                    <xs:enumeration
value="list"/>
                                  </xs:restriction>
                                </xs:simpleType>
```

```
                                </xs:attribute>
                                <xs:attribute name="Required"
type="xs:boolean"use="required"/>
                                <!--
                                        This
is the XPath of the element or attibute in the input
reference that will be storing the value
of this control
                                    -->
                                <xs:attribute name="XPath"
type="xs:string"use="required"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                        <!--
                            The attributes
listed below are used to identify the action to execute

in order to show the search criteria.
                        -->
                        <xs:attribute
name="MetadataScopeName"type="xs:string"use="optional"/>
                        <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                        <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                        <xs:attribute
name="ReferenceSchemaName"type="xs:string"use="optional"
/>
                        <xs:attribute
name="MetadataSolutionName"type="xs:string"use="optional"
/>
                        <xs:attribute name="ActionName"
type="xs:string"use="optional"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="Name" type="xs:string"
use="required"/>
                  <xs:attribute name="Caption"
type="xs:string"use="required"/>
                  <!--
                The attributes listed below
are used to identify the action to execute
```

```
                in order
to perform the search.
            -->
                    <xs:attribute name="MetadataScopeName"
type="xs:string"use="required"/>
                    <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ReferenceSchemaName"
type="xs:string"use="optional"/>
                    <xs:attribute name="MetadataSolutionName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ActionName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ActionType"
type="xs:string"use="optional"/>
                    <!--
                The Target attribute below can
be used to indicate how the results of the Search
action are displayed.  The default value is SearchRegion.
When Target="SearchRegion" the results of the action
will be displayed in one new region in the Search
Page.  Use this option in case only one region
is displayed as the result of executing the search action.
When Target="SearchPage" the results of the action will be
displayed in the Search Page.  Use this option in
case multiple regions are displayed ad the result of
executing the search action and the regions should appear in
the Search Page.  When Target="ReferencesPage" the results of
the action will be displayed in the References Page.
Use this option in case one or multiple regions are displaye
d as the result of executing the search action and the regio
ns should appear in the References Page.
            -->
                    <xs:attribute name="Target"
use="optional">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="SearchRegion"
/>
                          <xs:enumeration value="SearchPage"
/>
                          <xs:enumeration
```

```
value="ReferencesPage"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:attribute>
                  </xs:complexType>
                </xs:element>
                <!--
        Element:        HTML
        Description:
The HTML and XHTML elements below define the schema of the
xml
                        inside RegionProperties when invoking the
HTML region.
                        The element HTML can be used to pass
arbitrary HTML content
                        to the HTML region encoded
as xml string
    -->
                <xs:element name="HTML">
                  <xs:complexType>
                    <xs:simpleContent>
                      <xs:extension base="xs:string">
                        <xs:attributeGroup
ref="msibf:HtmlRegionAttributes"/>
                      </xs:extension>
                    </xs:simpleContent>
                  </xs:complexType>
                </xs:element>
                <!--
        Element:        XHTML
        Description:
The XHTML element can be used to pass arbitrary XHTML
content
                        to the HTML region
    -->
                <xs:element name="XHTML">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:any
namespace="http://www.w3.org/1999/xhtml"/>
                    </xs:sequence>
                    <xs:attributeGroup
ref="msibf:HtmlRegionAttributes"/>
                  </xs:complexType>
```

```
            </xs:element>
            <!--
        Attribute Group:  HtmlRegionAttributes

Description:     HtmlRegionAttributes contains a set of
attributes to be used in
                       conjunction with the
HTML of XHTML elements when rendering
                       inside of
an HTML region
    -->
            <xs:attributeGroup
name="HtmlRegionAttributes">
              <!--
            The Width attribute is used to
specify the width
            of the HTML display area of the HTML
region.  It has no effect when the
            region is displayed
in a page but it is considered when the region
            appears
in a dialog frame.
        -->
              <xs:attribute name="Width" use="optional">
                <xs:simpleType>
                  <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <!--
            The Height attribute is used to
specify the height
            of the HTML display area of the
HTML region
        -->
              <xs:attribute name="Height" use="optional">
                <xs:simpleType>
                  <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <!--
```

```
            The Scroll attribute is used to
indicate if the HTML display
            area of the HTML region
should have a scroll bar or not.
        -->
                <xs:attribute name="Scroll"
type="xs:boolean"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:
AssociationAttributes
        Description:        Extends
ReferenceAttributes
    -->
              <xs:attributeGroup
name="AssociationAttributes">
                <xs:attribute name="MetadataSolutionName"
type="xs:string"use="optional"/>
                <xs:attribute name="AssociationName"
type="xs:string"use="required"/>
                <xs:attribute name="Enabled"
type="xs:boolean"use="required"/>
                <xs:attribute name="MetadataScopeName"
type="xs:string"use="required"/>
                <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                <xs:attribute name="ReferenceSchemaName"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:    RegionAttributes

Description:        Every region must have these attributes,
even when displayed as a dialog.
    -->
              <xs:attributeGroup name="RegionAttributes">
                <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
                <xs:attribute name="RegionName"
type="xs:string"use="optional"/>
```

```
                <xs:attribute name="ShowAs" use="optional">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="MenuItem" />
                      <xs:enumeration value="ExpandedRegion"
/>
                      <xs:enumeration
value="CollapsedRegion"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
              </xs:attributeGroup>
              <!--
        Attribute Group:    ClassAttributes

Descripyion:        All the attributes, required to instantiate
a .NET class - except the
                            assembly.  The
operation should use the port to obtain the assembly.
    -->
              <xs:attributeGroup name="ClassAttributes">
                <xs:attribute name="TypeName"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:    CaptionAttributes

Description:        The caption is user visible string; the
description is a longer string
                            often used as a
tooltip.
    -->
              <xs:attributeGroup name="CaptionAttributes">
                <xs:attribute name="Caption"
type="xs:string"use="optional"/>
                <xs:attribute name="Description"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
            </xs:schema>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml" Name="SearchTypes"
IsCached="false"AuthenticationTypeValue="None">
          <Data>
```

```
            <Search
xmlns="http://schemas.microsoft.com/InformationBridge/2004">
              <SearchTypes>
              </SearchTypes>
            </Search>
          </Data>
        </Port>
      </Ports>
      <Operations>
        <Operation xsi:type="OperationCustom"
Name="ActivateRecognizer"TimeOut="0"
OperationTypeName="MSIBF.UI.ActivateRecognizer">
          <InputSchemas>
            <SchemaInstance
SchemaName="MSIBF.GenericRecognizer"Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="BuildDefaultSearchCriteriaRegion"EntityName="Search"
TimeOut="0"
TransformationName="BuildDefaultSearchCriteriaRegion">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="BuildSearchCriteriaRegion"EntityName="Search"
TimeOut="0"TransformationName="BuildSearchCriteriaRegion">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
            <SchemaInstance SchemaName="MSIBF.SearchType"
Index="2">
```

```
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="BuildSearchPickerRegion"EntityName="Search"
TimeOut="0"TransformationName="BuildSearchPickerRegion">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="BuildSearchSchema"EntityName="Search"TimeOut="0"
TransformationName="BuildSearchSchema">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="2">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationCustom"
Name="DeactivateRecognizer"TimeOut="0"
OperationTypeName="MSIBF.UI.DeactivateRecognizer">
```

```
          <InputSchemas>
            <SchemaInstance
SchemaName="MSIBF.GenericRecognizer"Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation xsi:type="OperationCustom"
Name="LoadActionHandler"TimeOut="0"
OperationTypeName="MSIBF.UI.LoadActionHandler"
PortName="GenericSmartTags">
          <InputSchemas>
            <SchemaInstance
SchemaName="MSIBF.GenericActionHandler"Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation xsi:type="OperationCustom"
Name="LoadRecognizer"TimeOut="0"
OperationTypeName="MSIBF.UI.LoadRecognizer"
PortName="GenericSmartTags">
          <InputSchemas>
            <SchemaInstance
SchemaName="MSIBF.GenericRecognizer"Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="LoadSearchFromSearchPort"EntityName="Search"
TimeOut="0">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="MSIBF.Search"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationQueryMenu"
Name="QuerySearchTypes"EntityName="Search"TimeOut="0"
QueryMenuTypeValue="Actions"MenuType="1">
```

```
            <OutputSchemas>
              <SchemaInstance SchemaName="MSIBF.Associations"
Index="1">
              </SchemaInstance>
            </OutputSchemas>
          </Operation>
          <Operation xsi:type="OperationCustom"
Name="ShowRegion"EntityName="Search"TimeOut="0"
OperationTypeName="MSIBF.UI.ShowRegion">
            <InputSchemas>
              <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
              </SchemaInstance>
            </InputSchemas>
          </Operation>
          <Operation
xsi:type="OperationTransformationAggregation"
Name="TransformationAggregation.LoadSearchType"
EntityName="Search"TimeOut="0">
            <InputSchemas>
              <SchemaInstance SchemaName="MSIBF.SearchType"
Index="1">
              </SchemaInstance>
            </InputSchemas>
            <OutputSchemas>
              <SchemaInstance SchemaName="MSIBF.SearchType"
Index="1">
              </SchemaInstance>
            </OutputSchemas>
          </Operation>
        </Operations>
        <Transformations>
          <Transformation xsi:type="TransformationXsl"
Name="BuildDefaultSearchCriteriaRegion"
InputSchemaName="MSIBF.Search"
OutputSchemaName="MSIBF.Region"
PortName="BuildDefaultSearchCriteriaRegionXslt"
ApplyTranslations="true">
          </Transformation>
          <Transformation xsi:type="TransformationXsl"
Name="BuildSearchCriteriaRegion"
InputSchemaName="MSIBF.Search"
OutputSchemaName="MSIBF.Region"
PortName="BuildSearchCriteriaRegionXslt"
```

```
ApplyTranslations="true">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="BuildSearchFromQueryMenu"
InputSchemaName="MSIBF.Associations"
OutputSchemaName="MSIBF.Search"
PortName="BuildSearchFromQueryMenuXslt"
ApplyTranslations="true">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="BuildSearchPickerRegion"
InputSchemaName="MSIBF.Search"
OutputSchemaName="MSIBF.Region"
PortName="BuildSearchPickerRegionXslt"
ApplyTranslations="true">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="BuildSearchResultsRegion"
InputSchemaName="MSIBF.SearchType"
OutputSchemaName="MSIBF.Region"
PortName="BuildSearchResultsRegionXslt"
ApplyTranslations="true">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="BuildSearchSchema"InputSchemaName="MSIBF.Search"
OutputSchemaName="MSIBF.Search"
PortName="BuildSearchSchemaXslt"ApplyTranslations="true">
        </Transformation>
        <Transformation
xsi:type="TransformationImmediatePort"
Name="GenericActionHandler"
InputSchemaName="MSIBF.GenericActionHandler"
OutputSchemaName="MSIBF.GenericActionHandler"
PortName="GenericActionHandler">
        </Transformation>
        <Transformation
xsi:type="TransformationImmediatePort"
Name="GenericRecognizer"
InputSchemaName="MSIBF.GenericRecognizer"
OutputSchemaName="MSIBF.GenericRecognizer"
PortName="GenericRecognizer">
        </Transformation>
        <Transformation
xsi:type="TransformationImmediatePort"
```

```
Name="LoadSearchFromSearchPort"
InputSchemaName="MSIBF.Search"
OutputSchemaName="MSIBF.Search"PortName="SearchTypes">
        </Transformation>
      </Transformations>
      <Schemas>
        <Schema xsi:type="SchemaXsd"
Name="MSIBF.Associations"PortName="MSIBF.Schemas"
Element="q1:Associations"
xmlns:q1="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaXsd"
Name="MSIBF.GenericActionHandler"PortName="MSIBF.Schemas"
Element="q2:GenericActionHandler"
xmlns:q2="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaXsd"
Name="MSIBF.GenericRecognizer"PortName="MSIBF.Schemas"
Element="q3:GenericRecognizer"
xmlns:q3="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="MSIBF.Region"
PortName="MSIBF.Schemas"Element="q4:Region"
xmlns:q4="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="MSIBF.Search"
PortName="MSIBF.Schemas"Element="q5:Search"
xmlns:q5="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="MSIBF.SearchType"
PortName="MSIBF.Schemas"Element="q6:SearchType"
xmlns:q6="http://schemas.microsoft.com/InformationBridge/200
4">
        </Schema>
        <Schema xsi:type="SchemaString"
Name="NullReference">
        </Schema>
      </Schemas>
      <Translations
```

```
xmlns:md="http://schemas.microsoft.com/InformationBridge/200
4/Metadata"
xmlns:mdd="http://schemas.microsoft.com/InformationBridge/20
04/MetadataDesigner/Schema">
        <Translation Key="SearchCriteriaRegionCaption"
Modifier="en-US"Value="Criteria"IsFallback="false">
        </Translation>
        <Translation Key="SearchCriteriaRegionDescription"
Modifier="en-US"Value="Criteria"IsFallback="false">
        </Translation>
        <Translation Key="SearchPickerRegionCaption"
Modifier="en-US"Value="Search Type"IsFallback="false">
        </Translation>
        <Translation Key="SearchPickerRegionDescription"
Modifier="en-US"Value="Search Type"IsFallback="false">
        </Translation>
        <Translation Key="SearchResultsRegionCaption"
Modifier="en-US"Value="Search Results"IsFallback="false">
        </Translation>
        <Translation Key="SearchResultsRegionDescription"
Modifier="en-US"Value="Search Results"IsFallback="false">
        </Translation>
      </Translations>
    </MetadataScope>
    <MetadataScope Name="Maconomy.SampleSolution.TS"
DisplayName="TimeSheet"Version="1.0.0.0"
DefaultEntityName="TimeSheet">
      <Entities>
        <Entity Name="TimeSheet"
DefaultReferenceSchemaName="AccessSchema"
DefaultViewName="LogonMaconomy">
          <Views>
            <View Name="LogonMaconomy" IsList="false">
              <MenuItemDefinition MenuType="1">
                <MenuDefinitionOperationInstance
OperationName="DefineMenuItemName"Index="0">
                  <TransformationInstances>
                    <TransformationInstance
TransformationName="MenuCaptionCreation"Index="1"
InputOperationIndex="-1001"
InputOperationOutputSchemaIndex="1">
                    </TransformationInstance>
                  </TransformationInstances>
                </MenuDefinitionOperationInstance>
```

```
                </MenuItemDefinition>
                <ViewLocators>
                  <ViewLocator
ReferenceSchemaName="AccessSchema">
                </ViewLocator>
                </ViewLocators>
                <Actions>
                  <Action Name="Logon Maconomy"
Type="EnterContext"IsSequentialExecution="true">
                    <MenuItemDefinition MenuType="0">
                    </MenuItemDefinition>
                    <OperationInstances>
                      <OperationInstance
OperationName="QueryMenu"Index="1">
                      </OperationInstance>
                      <OperationInstance
OperationName="ShowAssociations"Index="2">
                        <TransformationInstances>
                          <TransformationInstance Index="1"
InputOperationIndex="1"InputOperationOutputSchemaIndex="1">
                          </TransformationInstance>
                        </TransformationInstances>
                      </OperationInstance>
                      <OperationInstance
OperationName="ShowRegion"Index="3">
                        <TransformationInstances>
                          <TransformationInstance
TransformationName="LogOnRegionInput"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                          </TransformationInstance>
                        </TransformationInstances>
                      </OperationInstance>
                    </OperationInstances>
                  </Action>
                </Actions>
              </View>
              <View Name="TimeSheetDetails"
SchemaName="getResponse (maconomy.com)"IsList="false">
                <MenuItemDefinition MenuType="1">
                  <MenuDefinitionOperationInstance
OperationName="DefineMenuItemName"Index="0">
                </MenuDefinitionOperationInstance>
                </MenuItemDefinition>
                <ViewLocators>
```

```
                <ViewLocator ReferenceSchemaName="get
(maconomy.com)">
                  <OperationInstance OperationName="get"
Index="0">
                    <TransformationInstances>
                      <TransformationInstance Index="1"
InputOperationIndex="-1001"
InputOperationOutputSchemaIndex="1">
                      </TransformationInstance>
                    </TransformationInstances>
                  </OperationInstance>
                </ViewLocator>
              </ViewLocators>
              <Actions>
                <Action Name="Display Timesheet Details"
Type="EnterContext"IsSequentialExecution="true">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="ShowRegion"Index="1">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="Xsl.ShowRegion.Xsd.GetResponse"
Index="1"InputOperationIndex="-1"
InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
                <Action Name="Register Timesheet Line"
IsSequentialExecution="true">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="AddTimesheetLine"Index="1">
                      <TransformationInstances>
                        <TransformationInstance
TransformationName="AddLineRegionInput"Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
```

```
                    </OperationInstance>
                  </OperationInstances>
                </Action>
                <Action Name="Submit Timesheet Line"
IsSequentialExecution="false">
                  <MenuItemDefinition MenuType="0">
                  </MenuItemDefinition>
                  <OperationInstances>
                    <OperationInstance
OperationName="register"Index="1">
                      <TransformationInstances>
                        <TransformationInstance Index="1"
InputOperationIndex="0"InputOperationOutputSchemaIndex="1">
                        </TransformationInstance>
                      </TransformationInstances>
                    </OperationInstance>
                  </OperationInstances>
                </Action>
              </Actions>
            </View>
          </Views>
        </Entity>
      </Entities>
      <Ports>
        <Port xsi:type="PortFileXml" Name="AccessPort"
IsCached="false"AuthenticationTypeValue="Windows">
          <Data>
            <xs:schema targetNamespace="maconomy.com"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
              <xs:element name="Dialog">
                <xs:complexType>
                  <xs:attribute name="name" type="xs:string"
use="required">
                  </xs:attribute>
                </xs:complexType>
              </xs:element>
            </xs:schema>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="AddLineRegionInput"IsCached="false"
AuthenticationTypeValue="Windows">
          <Data>
```

```
            <msibf:Region Enabled="true"
EntityName="TimeSheet"
MetadataScopeName="Maconomy.SampleSolution.TS"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
              <msibf:RegionProperties Caption="Add a
timesheet line" Description="Add a timesheet line"
TypeName="Maconomy.Sample.UserInterface.RegisterLine"
ShowAs="ExpandedRegion">
              </msibf:RegionProperties>
            </msibf:Region>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml" Name="LogOnRegionInput"
IsCached="false"AuthenticationTypeValue="Windows">
          <Data>
            <msibf:Region Enabled="true"
EntityName="TimeSheet"
MetadataScopeName="Maconomy.SampleSolution.TS"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
              <msibf:RegionProperties Caption="Maconomy
Access Control" Description="Log On Region"
TypeName="Maconomy.Sample.UserInterface.LogOn"
ShowAs="ExpandedRegion">
              </msibf:RegionProperties>
            </msibf:Region>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="MenuDefinitionPort"IsCached="false"
AuthenticationTypeValue="Windows">
          <Data>
            <xs:element name="ViewProperties"
type="ViewPropertiesType"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
              <xs:complexType name="ViewPropertiesType">
                <xs:sequence>
                  <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="Caption"
type="xs:string"use="optional"default=""/>
                <xs:attribute name="Description"
```

```
type="xs:string"use="optional"default=""/>
                </xs:complexType>
              </xs:element>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="MenuDefinitionProvider"IsCached="false"
AuthenticationTypeValue="Windows">
            <Data>
              <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:iwb="http://schemas.microsoft.com/InformationBridge/20
04">
                <xsl:template match="/">
                  <iwb:ViewProperties Description="Default
TimeSheet view" ShowAs="MenuItem">
                    <xsl:attribute name="Caption">Maconomy
<xsl:value-of
select="//@Name"></xsl:value-of></xsl:attribute>
                  </iwb:ViewProperties>
                </xsl:template>
              </xsl:stylesheet>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml" Name="MSIBF.Schemas"
IsCached="false"AuthenticationTypeValue="Windows">
            <Data>
              <xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified"
targetNamespace="http://schemas.microsoft.com/InformationBri
dge/2004"xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                <!--
          Root Element:   Associations

Operation:      QueryMenu, Custom/ShowAssociations.

Commands:       QueryMenu, Navigation.
          Description:    The
QueryMenu schema is UI agnostic; the ShowAssociations schema
is its
                          extension.  It is easier to read and
write the shared schema then to use
```

```
                        extensions
or substitution groups, but some parts of it might not be
ever used/generated by the QueryMenu operation.

-->
              <xs:element name="Associations">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Properties">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element
ref="msibf:ViewProperties"minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:choice minOccurs="0"
maxOccurs="unbounded">
                      <xs:element ref="msibf:Action" />
                      <xs:element ref="msibf:Region" />
                      <xs:element ref="msibf:Relationship"
/>
                      <xs:element ref="msibf:ViewReference"
/>
                    </xs:choice>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   Region
        Operation:
Custom/ShowRegion, Custom/ShowDialog
        Descripyion:    The
region can appear as a part of the associations sequence or
standalone.
    -->
              <xs:element name="Region"
type="msibf:AssociationType"/>
              <xs:element name="Relationship"
type="msibf:AssociationType"/>
              <xs:element name="ViewReference"
type="msibf:AssociationType"/>
              <xs:element name="Action">
                <xs:complexType>
```

```
                    <xs:complexContent mixed="true">
                      <xs:extension
base="msibf:AssociationType">
                        <xs:attribute name="ActionType" />
                      </xs:extension>
                    </xs:complexContent>
                  </xs:complexType>
                </xs:element>
                <!--
        Root Element:   Properties
        Operation:
ShowAssociations
        Descripyion:    Description for the pane
that shows associations
    -->
                <xs:element name="ViewProperties"
type="msibf:ViewPropertiesType"/>
                <!--
        Root Element:   Extension
        Operation:
Custom/LoadExtension
    -->
                <xs:element name="Extension">
                  <xs:complexType>
                    <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                  </xs:complexType>
                </xs:element>
                <!--
        Root Element:   Text
        Operation:
Custom/InsertText
    -->
                <xs:element name="Text" type="xs:string" />
                <!--
        Root Element:   Message
        Operation:
Custom/ShowMessage
    -->
                <xs:element name="Message">
                  <xs:complexType>
                    <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
                  </xs:complexType>
```

```
            </xs:element>
            <!--
        Root Element:   AnnotatedReferences

Operation:      Custom/InsertResults
        Description:
Annotated references wraps multiple AnnotatedReference
elements
    -->
            <xs:element name="AnnotatedReferences">
              <xs:complexType>
                <xs:sequence>
                  <xs:element
ref="msibf:AnnotatedReference"minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <!--
        Root Element:   AnnotatedReference

Operation:      Custom/InsertReference, Custom/ShowContext

Description:    Annotated reference wraps the reference
itself, together with caption
                        attributes and
optional text.  The name of the reference element is not

known, but its position does not ever change.
    -->
            <xs:element name="AnnotatedReference">
              <xs:complexType>
                <xs:sequence>
                  <!-- Substitution is required for this
to validate the abstract reference -->
                  <xs:element
ref="msibf:AbstractReference"/>
                  <xs:element ref="msibf:Text"
minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
              </xs:complexType>
            </xs:element>
```

```
            <!--
        Root Element:   GenericRecognizer

Operation:      Custom/LoadRecognizer,
Custom/ActivateRecognizer, Custom/DeactivateReconizer

Description:    Describes a generic recognizer.  When calling
into the Activate/Deactivate
                        methods, the ID
attribute is the only one used - the other nodes are
ignored.
                        The arbitrary element is passed to the
Initialize method of the IRecognizer interface.
    -->
              <xs:element name="GenericRecognizer">
                <xs:complexType>
                  <xs:sequence>
                    <xs:any minOccurs="0" maxOccurs="1" />
                  </xs:sequence>
                  <xs:attribute default="" name="Id"
type="xs:string"/>
                  <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                </xs:complexType>
              </xs:element>
              <!--
        Root Element:   GenericActionHandler

Operation:      Custom/LoadActionHandler
        Description:
Describes a generic action handler.  The arbitrary element
is passed to the Initialize
                        method of the
IActionHandler interface.
    -->
              <xs:element name="GenericActionHandler">
                <xs:complexType>
                  <xs:sequence>
                    <xs:any minOccurs="0" maxOccurs="1" />
                  </xs:sequence>
                  <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                </xs:complexType>
              </xs:element>
```

```
            <!--
        Operation:      Custom/GetProperties

Description:    Describes the properties requested and
returned from the MSIBF.UI.GetProperties custom operation
type.
    -->
            <xs:element name="Properties">
              <xs:complexType>
                <xs:sequence minOccurs="0"
maxOccurs="unbounded">
                  <xs:element name="Property">
                    <xs:complexType>
                      <xs:attribute name="Name"
type="xs:string"/>
                      <xs:attribute name="Value"
type="xs:string"/>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <!--
        Element:        AbstractReference

Description:    The global element defining the substitution
group.
                        Because it is abstract, it cannot appear
in the XML and it needs substitution.
    -->
            <xs:element name="AbstractReference"
type="msibf:AbstractReferenceType"/>
            <!--
        Complex Type:   ViewPropertiesType

Description:    UI hints for the pane that hosts associations.
    The ShowAssociations operation assumes (but does
not require) the presence of
                        the UI hints, but
otherwise the schema of the content is open.
    -->
            <xs:complexType name="ViewPropertiesType">
              <xs:sequence>
                <xs:any namespace="##any"
```

```
processContents="lax"minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="Caption"
type="xs:string"use="optional"default=""/>
                <xs:attribute name="Description"
type="xs:string"use="optional"default=""/>
              </xs:complexType>
              <!--
        Complex Type:   AssociationType

Description:    Each association must have the attributes as
defined by the QueryMenu operation.
                        The
ShowAssociations operation assumes (but does not require)
the presence of
                        the UI hints, but otherwise the
schema of the content is open.
    -->
              <xs:complexType name="AssociationType"
mixed="true">
                <xs:sequence>
                  <!-- The data to be passed to the region
may contain and
                and must contain the reference.

Substitution is required for this to validate the abstract
reference.
            -->
                  <xs:element ref="msibf:AbstractReference"
/>
                  <!-- The data to be passed to the region
may contain properties of type RegionPropertiesType -->
                  <xs:element ref="msibf:RegionProperties"
minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup
ref="msibf:AssociationAttributes"/>
              </xs:complexType>
              <!--
        Complex Type:   AbstractReferenceType

Description:    A view reference type defined as open, so that
it can contain any elements
                        and attributes.  This
```

```
might not be suitable in general, so it cannot be used

as the base type for references.


Since this type is abstract it cannot be istantiated but a
XML element instance
                        can be substituted with this
type.  An example of a reference and it's substitution:

<xs:element name="myReference"

substitutionGroup="msibf:AbstractReference"
type="msibf:AbstractReferenceType" />

<myReference msibf:MetadataScopeName="..." ...  />
    -->
              <xs:complexType name="AbstractReferenceType"
mixed="true">
                <xs:complexContent>
                  <xs:extension
base="msibf:AbstractReferenceAttributesType">
                    <xs:sequence>
                      <xs:any minOccurs="0"
maxOccurs="unbounded"namespace="##any"
processContents="lax"/>
                    </xs:sequence>
                    <xs:anyAttribute namespace="##any"
processContents="lax"/>
                  </xs:extension>
                </xs:complexContent>
              </xs:complexType>
              <!--
        Complex Type:   RegionPropertiesType

Description:    Some region properties are predefined, but
additional attributes might be used.
    -->
              <xs:complexType name="RegionPropertiesType">
                <xs:sequence>
                  <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup
```

```
ref="msibf:RegionAttributes"/>
                <xs:anyAttribute namespace="##any"
processContents="lax"/>
            </xs:complexType>
            <xs:element name="RegionProperties"
type="msibf:RegionPropertiesType"/>
            <!--
        Complex Type:
AbstractReferenceAttributesType
        Description:    It is not
possible in XSD to define the attributes for an element
without
                        specifying the element name.  This is a
problem for view references - the
                        solution is to
use substitution groups.

                        The abstract
reference type defines the common subtype - all the elements

in the substitution group must use a same type
or a derived type.
                        The elements in the
substitution group could use any element name.
    -->
            <xs:complexType
name="AbstractReferenceAttributesType"abstract="true">
                <xs:attribute name="MetadataScopeName"
form="qualified"type="xs:string"use="required"/>
                <xs:attribute name="EntityName"
form="qualified"type="xs:string"use="optional"/>
                <xs:attribute name="ViewName"
form="qualified"type="xs:string"use="optional"/>
                <xs:attribute name="ReferenceSchemaName"
form="qualified"type="xs:string"use="optional"/>
            </xs:complexType>
            <!--
        Element:        Credentials

Description:    This is the schema of the authentication data
that
                        the
AuthenticationDefinitionOperationInstance of a Port must
comply with.
```

```
    -->
              <xs:element name="Credentials">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="UserName"
type="xs:string"/>
                    <xs:element name="Password"
type="xs:string"/>
                    <xs:element name="DomainName"
type="xs:string"minOccurs="0"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
              <!--
        Element:        Search
        Description:
This is the schema of the metadata for Search.

It contains a list of search types.
    -->
              <xs:element name="Search">
                <xs:complexType>
                  <xs:sequence>
                    <!-- This is the list of search types
(actions) available for the solution -->
                    <xs:element name="SearchTypes">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element ref="msibf:SearchType"
minOccurs="0"maxOccurs="unbounded"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <!--
        Element:        SearchType

Description:    This is the schema of the one search type.

A search type is used to define one individual search
action,
                        the associated reference used to invoke
```

```
the action,
                        and the search criteria UI used to
collect user input.
    -->
              <xs:element name="SearchType">
                <xs:complexType>
                  <xs:sequence>
                    <!--
                This is the xml framgent
of the search criteria.
                It gets filed with user data
and is used as reference for the search action invocation.

-->
                    <xs:element
name="InputReferenceInstance"minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:any namespace="##any"
processContents="lax"minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="Namespaces"
use="required"/>
                      </xs:complexType>
                    </xs:element>
                    <!--
                This is the list of search
criteria UI elements that persist data in the

reference instance defined above.
                -->
                    <xs:element name="SearchCriteria"
minOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <!-- This is the definition of one
input control -->
                          <xs:element name="SearchCriterion"
minOccurs="0"maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <!-- This is the list of
possible values that
                                        pre-populate the
```

```
list input control -->
                                    <xs:element name="Item"
minOccurs="0"maxOccurs="unbounded">
                                      <xs:complexType>
                                        <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                            <xs:attribute
name="Value"type="xs:string"use="optional"/>
                                            <xs:attribute
name="Selected"type="xs:boolean"use="optional"/>
                                          </xs:extension>
                                        </xs:simpleContent>
                                      </xs:complexType>
                                    </xs:element>
                                  </xs:sequence>
                                  <xs:attribute name="Caption"
type="xs:string"use="required"/>
                                  <xs:attribute
name="Description"type="xs:string"use="required"/>
                                  <xs:attribute name="Type"
use="required">
                                    <xs:simpleType>
                                      <xs:restriction
base="xs:string">
                                        <xs:enumeration
value="text"/>
                                        <xs:enumeration
value="list"/>
                                      </xs:restriction>
                                    </xs:simpleType>
                                  </xs:attribute>
                                  <xs:attribute name="Required"
type="xs:boolean"use="required"/>
                                  <!--
                                        This
is the XPath of the element or attibute in the input
reference
    that will be storing the value
of this control
                                    -->
                                  <xs:attribute name="XPath"
type="xs:string"use="required"/>
                                </xs:complexType>
```

```
                        </xs:element>
                    </xs:sequence>
                    <!--
                        The attributes
listed below are used to identify the action to execute

in order to show the search criteria.
                    -->
                    <xs:attribute
name="MetadataScopeName"type="xs:string"use="optional"/>
                    <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                    <xs:attribute
name="ReferenceSchemaName"type="xs:string"use="optional"
/>
                    <xs:attribute
name="MetadataSolutionName"type="xs:string"use="optional"
/>
                    <xs:attribute name="ActionName"
type="xs:string"use="optional"/>
                      </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="Name" type="xs:string"
use="required"/>
                <xs:attribute name="Caption"
type="xs:string"use="required"/>
                <!--
              The attributes listed below
are used to identify the action to execute
              in order
to perform the search.
            -->
                <xs:attribute name="MetadataScopeName"
type="xs:string"use="required"/>
                <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                <xs:attribute name="ReferenceSchemaName"
type="xs:string"use="optional"/>
                <xs:attribute name="MetadataSolutionName"
```

```
type="xs:string"use="optional"/>
                    <xs:attribute name="ActionName"
type="xs:string"use="optional"/>
                    <xs:attribute name="ActionType"
type="xs:string"use="optional"/>
                    <!--
                The Target attribute below can
be used to indicate how the results of the
                Search
action are displayed.  The default value is SearchRegion.

When Target="SearchRegion" the results of the action
will be displayed in one
                new region in the Search
Page.  Use this option in case only one region is displayed

as the result of executing the search action.


When Target="SearchPage" the results of the action will be
displayed in the Search
                Page.  Use this option in
case multiple regions are displayed ad the result of

executing the search action and the regions should appear in
the Search Page.

                When
Target="ReferencesPage" the results of the action will be
displayed in the References
                Page.  Use this option in
case one or multiple regions are displayed as the result of

executing the search action and the regions should appear in
the References Page.
            -->
                    <xs:attribute name="Target"
use="optional">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:enumeration value="SearchRegion"
/>
                          <xs:enumeration value="SearchPage"
/>
```

```
                        <xs:enumeration
value="ReferencesPage"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:attribute>
                </xs:complexType>
              </xs:element>
              <!--
        Element:        HTML
        Description:
The HTML and XHTML elements below define the schema of the
xml
                        inside RegionProperties when invoking the
HTML region.
                        The element HTML can be used to pass
arbitrary HTML content
                        to the HTML region encoded
as xml string
    -->
              <xs:element name="HTML">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attributeGroup
ref="msibf:HtmlRegionAttributes"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <!--
        Element:        XHTML
        Description:
The XHTML element can be used to pass arbitrary XHTML
content
                        to the HTML region
    -->
              <xs:element name="XHTML">
                <xs:complexType>
                  <xs:sequence>
                    <xs:any
namespace="http://www.w3.org/1999/xhtml"/>
                  </xs:sequence>
                  <xs:attributeGroup
ref="msibf:HtmlRegionAttributes"/>
```

```
                </xs:complexType>
              </xs:element>
              <!--
        Attribute Group:  HtmlRegionAttributes

Description:     HtmlRegionAttributes contains a set of
attributes to be used in
                         conjunction with the
HTML of XHTML elements when rendering
                         inside of
an HTML region
    -->
              <xs:attributeGroup
name="HtmlRegionAttributes">
                <!--
            The Width attribute is used to
specify the width
            of the HTML display area of the HTML
region.  It has no effect when the
            region is displayed
in a page but it is considered when the region
            appears
in a dialog frame.
        -->
                <xs:attribute name="Width" use="optional">
                  <xs:simpleType>
                    <xs:restriction base="xs:integer">
                      <xs:minInclusive value="0" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
                <!--
            The Height attribute is used to
specify the height
            of the HTML display area of the
HTML region
        -->
                <xs:attribute name="Height" use="optional">
                  <xs:simpleType>
                    <xs:restriction base="xs:integer">
                      <xs:minInclusive value="0" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
```

```
                <!--
            The Scroll attribute is used to
indicate if the HTML display
            area of the HTML region
should have a scroll bar or not.
        -->
                <xs:attribute name="Scroll"
type="xs:boolean"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:
AssociationAttributes
        Description:        Extends
ReferenceAttributes
    -->
              <xs:attributeGroup
name="AssociationAttributes">
                <xs:attribute name="MetadataSolutionName"
type="xs:string"use="optional"/>
                <xs:attribute name="AssociationName"
type="xs:string"use="required"/>
                <xs:attribute name="Enabled"
type="xs:boolean"use="required"/>
                <xs:attribute name="MetadataScopeName"
type="xs:string"use="required"/>
                <xs:attribute name="EntityName"
type="xs:string"use="optional"/>
                <xs:attribute name="ViewName"
type="xs:string"use="optional"/>
                <xs:attribute name="ReferenceSchemaName"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:    RegionAttributes

Description:        Every region must have these attributes,
even when displayed as a dialog.
    -->
              <xs:attributeGroup name="RegionAttributes">
                <xs:attributeGroup
ref="msibf:ClassAttributes"/>
                <xs:attributeGroup
ref="msibf:CaptionAttributes"/>
                <xs:attribute name="RegionName"
```

```
type="xs:string"use="optional"/>
                <xs:attribute name="ShowAs" use="optional">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="MenuItem" />
                      <xs:enumeration value="ExpandedRegion"
/>
                      <xs:enumeration
value="CollapsedRegion"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:attribute>
              </xs:attributeGroup>
              <!--
        Attribute Group:    ClassAttributes

Descripyion:        All the attributes, required to instantiate
a .NET class - except the
                            assembly.  The
operation should use the port to obtain the assembly.
    -->
              <xs:attributeGroup name="ClassAttributes">
                <xs:attribute name="TypeName"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
              <!--
        Attribute Group:    CaptionAttributes

Description:        The caption is user visible string; the
description is a longer string
                            often used as a
tooltip.
    -->
              <xs:attributeGroup name="CaptionAttributes">
                <xs:attribute name="Caption"
type="xs:string"use="optional"/>
                <xs:attribute name="Description"
type="xs:string"use="optional"/>
              </xs:attributeGroup>
            </xs:schema>
          </Data>
        </Port>
        <Port xsi:type="PortFileXml"
Name="RegionInputTransformPort"IsCached="false"
```

```
AuthenticationTypeValue="Windows">
        </Port>
        <Port xsi:type="PortSoap"
Name="Soap.timesheetSOAPPort"
Location="http://cph-pc318:19001/cgi-bin/Maconomy/MaconomyWS
.ibf.W_MCS.exe/soap.ms"IsCached="false"
AuthenticationTypeValue="Windows">
        </Port>
        <Port xsi:type="PortAssembly" Name="UIAssembly"
Location="D:\Thesis\Sample\UserInterface\bin\Debug\UserInter
face.dll"IsCached="true"AuthenticationTypeValue="Windows">
        </Port>
        <Port xsi:type="PortFileXml" Name="Xsd.maconomy.com"
IsCached="false"AuthenticationTypeValue="None">
          <Data>
            <schema targetNamespace="maconomy.com"
xmlns="http://www.w3.org/2001/XMLSchema">
              <simpleType name="amount">
                <restriction base="double" />
              </simpleType>
              <element name="sessionid" type="string" />
              <element name="Security">
                <complexType>
                  <sequence>
                    <element name="UsernameToken">
                      <complexType>
                        <sequence>
                          <element minOccurs="1"
maxOccurs="1"name="Username"type="string"/>
                        </sequence>
                      </complexType>
                    </element>
                  </sequence>
                </complexType>
              </element>
              <complexType name="getreferType">
                <sequence>
                  <element minOccurs="1" maxOccurs="1"
name="username"type="string"/>
                  <element minOccurs="1" maxOccurs="1"
name="password"type="string"/>
                  <element minOccurs="1" maxOccurs="1"
name="PeriodStart"type="date"/>
                </sequence>
```

```
            </complexType>
            <element name="get">
              <complexType>
                <sequence>
                  <element name="refer"
type="q1:getreferType"xmlns:q1="maconomy.com"/>
                </sequence>
              </complexType>
            </element>
            <complexType name="getResponseType0">
              <sequence>
                <element minOccurs="1" maxOccurs="1"
name="JobNumber"type="string"/>
                <element minOccurs="1" maxOccurs="1"
name="ActivityNumber"type="string"/>
                <element minOccurs="1" maxOccurs="1"
name="TaskName"type="string"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay1"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay2"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay3"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay4"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay5"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay6"type="double"/>
                <element minOccurs="1" maxOccurs="1"
name="NumberOfDay7"type="double"/>
              </sequence>
            </complexType>
            <complexType name="getResponseType1">
              <sequence>
                <element minOccurs="0"
maxOccurs="unbounded"name="element"
type="q2:getResponseType0"xmlns:q2="maconomy.com"/>
              </sequence>
            </complexType>
            <complexType name="getResponseType">
              <sequence>
                <element minOccurs="1" maxOccurs="1"
name="EmployeeNumber"type="string"/>
```

```
                    <element minOccurs="1" maxOccurs="1"
name="PeriodStart"type="date"/>
                    <element minOccurs="1" maxOccurs="1"
name="Approved"type="boolean"/>
                    <element minOccurs="1" maxOccurs="1"
name="Rows"type="q3:getResponseType1"
xmlns:q3="maconomy.com"/>
                  </sequence>
                </complexType>
                <element name="getResponse"
type="q4:getResponseType"xmlns:q4="maconomy.com"/>
                <complexType name="registerreferType">
                  <sequence>
                    <element minOccurs="1" maxOccurs="1"
name="username"type="string"/>
                    <element minOccurs="1" maxOccurs="1"
name="password"type="string"/>
                    <element minOccurs="1" maxOccurs="1"
name="PeriodStart"type="date"/>
                  </sequence>
                </complexType>
                <complexType name="registerentryType">
                  <sequence>
                    <element minOccurs="1" maxOccurs="1"
name="JobNumber"type="string"/>
                    <element minOccurs="1" maxOccurs="1"
name="ActivityNumber"type="string"/>
                    <element minOccurs="1" maxOccurs="1"
name="TaskName"type="string"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay1"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay2"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay3"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay4"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay5"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay6"type="double"/>
                    <element minOccurs="1" maxOccurs="1"
name="NumberOfDay7"type="double"/>
                  </sequence>
```

```
                </complexType>
                <element name="register">
                  <complexType>
                    <sequence>
                      <element name="refer"
type="q5:registerreferType"xmlns:q5="maconomy.com"/>
                      <element name="entry"
type="q6:registerentryType"xmlns:q6="maconomy.com"/>
                    </sequence>
                  </complexType>
                </element>
                <element name="registerResponse"
type="boolean"/>
              </schema>
            </Data>
          </Port>
          <Port xsi:type="PortFileXml"
Name="Xsl.ShowRegion.Xsd.GetResponse"IsCached="false"
AuthenticationTypeValue="Windows">
            <Data>
              <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:iwb="http://schemas.microsoft.com/InformationBridge/20
04/Translator">
                <xsl:template match="/">
                  <msibf:Region Enabled="true"
EntityName="TimeSheet"
MetadataScopeName="Maconomy.SampleSolution.TS"
xmlns:msibf="http://schemas.microsoft.com/InformationBridge/
2004">
                    <msibf:RegionProperties Caption="Maconomy
Time Sheets" Description="Display Maconomy timesheets
details"
TypeName="Maconomy.Sample.UserInterface.DisplayTimesheet"
ShowAs="ExpandedRegion">
                      <xsl:copy-of select="/" />
                    </msibf:RegionProperties>
                  </msibf:Region>
                </xsl:template>
              </xsl:stylesheet>
            </Data>
          </Port>
        </Ports>
        <Operations>
```

```
        <Operation xsi:type="OperationCustom"
Name="AddTimesheetLine"TimeOut="20"
OperationTypeName="MSIBF.UI.ShowDialog"
PortName="UIAssembly">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation
xsi:type="OperationTransformationAggregation"
Name="DefineMenuItemName"EntityName="TimeSheet"
TimeOut="0">
          <InputSchemas>
            <SchemaInstance
SchemaName="MenuDefinitionSchema"Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance
SchemaName="MenuDefinitionSchema"Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationSoapRequest"
Name="get"EntityName="TimeSheet"TimeOut="10"
IsUpdate="false"PortName="Soap.timesheetSOAPPort"
SoapAction="timesheet">
          <InputSchemas>
            <SchemaInstance SchemaName="get (maconomy.com)"
Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="getResponse
(maconomy.com)" Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationQueryMenu"
Name="QueryMenu"TimeOut="0"QueryMenuTypeValue=""
MenuType="1">
          <OutputSchemas>
```

```
            <SchemaInstance SchemaName="MSIBF.Associations"
Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationSoapRequest"
Name="register"EntityName="TimeSheet"TimeOut="20"
IsUpdate="false"PortName="Soap.timesheetSOAPPort"
SoapAction="timesheet">
          <InputSchemas>
            <SchemaInstance SchemaName="register
(maconomy.com)" Index="1">
            </SchemaInstance>
          </InputSchemas>
          <OutputSchemas>
            <SchemaInstance SchemaName="registerResponse
(maconomy.com)" Index="1">
            </SchemaInstance>
          </OutputSchemas>
        </Operation>
        <Operation xsi:type="OperationCustom"
Name="ShowAssociations"TimeOut="0"
OperationTypeName="MSIBF.UI.ShowAssociations">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Associations"
Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
        <Operation xsi:type="OperationCustom"
Name="ShowRegion"TimeOut="0"
OperationTypeName="MSIBF.UI.ShowRegion"
PortName="UIAssembly">
          <InputSchemas>
            <SchemaInstance SchemaName="MSIBF.Region"
Index="1">
            </SchemaInstance>
          </InputSchemas>
        </Operation>
      </Operations>
      <Transformations>
        <Transformation
xsi:type="TransformationImmediatePort"
Name="AddLineRegionInput"OutputSchemaName="MSIBF.Region"
```

```
PortName="AddLineRegionInput">
        </Transformation>
        <Transformation
xsi:type="TransformationImmediatePort"
Name="LogOnRegionInput"OutputSchemaName="MSIBF.Region"
PortName="LogOnRegionInput">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="MenuCaptionCreation"InputSchemaName="AccessSchema"
OutputSchemaName="MenuDefinitionSchema"
PortName="MenuDefinitionProvider"ApplyTranslations="false">
        </Transformation>
        <Transformation xsi:type="TransformationXsl"
Name="Xsl.ShowRegion.Xsd.GetResponse"
InputSchemaName="getResponse (maconomy.com)"
OutputSchemaName="MSIBF.Region"
PortName="Xsl.ShowRegion.Xsd.GetResponse"
ApplyTranslations="true">
        </Transformation>
      </Transformations>
      <Schemas>
        <Schema xsi:type="SchemaXsd" Name="AccessSchema"
PortName="AccessPort"Element="q7:Dialog"
xmlns:q7="maconomy.com">
        </Schema>
        <Schema xsi:type="SchemaString" Name="AnySchema">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="get
(maconomy.com)" PortName="Xsd.maconomy.com" Element="q8:get"
xmlns:q8="maconomy.com">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="getResponse
(maconomy.com)" PortName="Xsd.maconomy.com"
Element="q9:getResponse"xmlns:q9="maconomy.com">
        </Schema>
        <Schema xsi:type="SchemaXsd"
Name="MenuDefinitionSchema"PortName="MenuDefinitionPort">
        </Schema>
        <Schema xsi:type="SchemaXsd"
Name="MSIBF.Associations"PortName="MSIBF.Schemas"
Element="q10:Associations"
xmlns:q10="http://schemas.microsoft.com/InformationBridge/20
04">
        </Schema>
```

```
        <Schema xsi:type="SchemaXsd" Name="MSIBF.Region"
PortName="MSIBF.Schemas"Element="q11:Region"
xmlns:q11="http://schemas.microsoft.com/InformationBridge/20
04">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="register
(maconomy.com)" PortName="Xsd.maconomy.com"
Element="q12:register"xmlns:q12="maconomy.com">
        </Schema>
        <Schema xsi:type="SchemaXsd" Name="registerResponse
(maconomy.com)" PortName="Xsd.maconomy.com"
Element="q13:registerResponse"xmlns:q13="maconomy.com">
        </Schema>
      </Schemas>
    </MetadataScope>
  </MetadataScopes>
  <OperationTypes
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <OperationType Name="MSIBF.UI.ActivateRecognizer">
    </OperationType>
    <OperationType Name="MSIBF.UI.DeactivateRecognizer">
    </OperationType>
    <OperationType Name="MSIBF.UI.LoadActionHandler">
    </OperationType>
    <OperationType Name="MSIBF.UI.LoadRecognizer">
    </OperationType>
    <OperationType Name="MSIBF.UI.ShowAssociations">
    </OperationType>
    <OperationType Name="MSIBF.UI.ShowDialog">
    </OperationType>
    <OperationType Name="MSIBF.UI.ShowRegion">
    </OperationType>
  </OperationTypes>
  <Timestamp
RetrieveTime="2007-03-21T17:12:31.0770000+01:00"
Timestamp="2007-03-21T15:56:44.0430000+01:00"
xmlns:md="http://schemas.microsoft.com/InformationBridge/200
4/Metadata"
xmlns:mdd="http://schemas.microsoft.com/InformationBridge/20
04/MetadataDesigner/Schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <MetadataScopeTimestamps>
      <MetadataScopeTimestamp
```

```
MetadataScopeName="InformationBridge"
Timestamp="2007-03-19T10:09:36.4500000+01:00">
        <EntityTimestamps>
          <EntityTimestamp EntityName="GenericSmartTags"
Timestamp="2007-03-17T14:14:34.2630000+01:00">
          </EntityTimestamp>
          <EntityTimestamp EntityName="Search"
Timestamp="2007-03-17T14:14:34.2630000+01:00">
          </EntityTimestamp>
        </EntityTimestamps>
      </MetadataScopeTimestamp>
      <MetadataScopeTimestamp
MetadataScopeName="Maconomy.SampleSolution.TS"
Timestamp="2007-03-21T17:12:30.9370000+01:00">
        <EntityTimestamps>
          <EntityTimestamp EntityName="TimeSheet"
Timestamp="2007-03-21T17:12:30.9370000+01:00">
          </EntityTimestamp>
        </EntityTimestamps>
      </MetadataScopeTimestamp>
    </MetadataScopeTimestamps>
  </Timestamp>
</Metadata>
```

Appendix E

Source code of UI components

This section contains all of the source codes developped in Visual Studio for the UI components of the prototype solution.

E.1 SmartTag Assembly

The following file contains the definition for the Cusomer Action Handler discussed in section 4.6.

TimeSheetActionHandler.cs

```
/*============================================================
===============
  This file is part of Maconomy IBF-based TimeSheets Sample.

©Author Xuesong Liu,     All rights reserved.

============================================================
```

```
==============*/
using System;
using System.IO;
using System.ComponentModel;
using System.Globalization;
using System.Text.RegularExpressions;
using System.Threading;
using System.Xml;

using Microsoft.InformationBridge.Framework.Interfaces;
using Microsoft.InformationBridge.Framework.UI.Interop;
using Microsoft.Office.Interop.SmartTag;

namespace Maconomy.Sample.SmartTag
{
    /// <summary>
    /// Summary description for TimeSheetActionHandler.
    /// </summary>
    public class TimeSheetActionHandler : IActionHandler
    {
        public TimeSheetActionHandler()
        { }

        protected IActionHandlerMenuItem ShowTimeSheetsMenu
        {
            get { return new ShowTimeSheetsMenuItem(); }
        }

        #region Implementation Members
        /// <summary>
        /// Executes a show timesheets action against IBF.
        /// </summary>
        /// <param name="mediator">IMediator that provides
access to the Host App.</param>
        /// <param name="context">Context information for the
current context in IBF.</param>
        protected virtual void ShowTimeSheets ( IMediator
mediator, IContextInformation context )
        {


/*context.MetadataScopeName="Maconomy.SampleSolution.TS";
            StreamWriter sw =
File.CreateText("d:/Thesis/newFile.txt");
```

```
            sw.Write(context.MetadataScopeName);
            sw.Close();*/

            // Execute the hardcoded action - "Show Timesheets".
            IContextFactory2 factory = Facade.ContextFactory as
IContextFactory2;

            if ( factory.IsMetadataScopeDefined(
context.MetadataScopeName ))
            {
                IEngineCommand command =
mediator.CreateLocalCommand( "MSIBF.UI.ShowContext",
context.Reference );
                command.Context = context;
                command.Execute();

            }
        }
        #endregion

        #region IActionHandler Members
        /// <summary>
        /// Returns an array of valid menu items.
        /// </summary>
        /// <param name="mediator">IMediator to be used to
communicate with the host applicationparam>
        /// <param name="properties">ISmartTagProperties
associated with the recognized term</param>
        /// <returns>An array of valid menu items to expose this
Action Handler's functionality</returns>
        public virtual IActionHandlerMenuItem[] GetMenuItems(
IMediator mediator, ISmartTagProperties properties )
        {
            IActionHandlerMenuItem[] menuItems = new
IActionHandlerMenuItem[1] { this.ShowTimeSheetsMenu };
            return menuItems;
        }

        /// <summary>
        /// Used to setup the action handler.
        /// </summary>
        /// <param name="initializationData">XmlElement
containing the data that helps to define this
        /// action handler.</param>
```

```
        /// <remarks>In this implementation, this method is
effectively ignored because there is
        /// nothing to initialize.</remarks>
        public virtual void Initialize( XmlElement
initializationData )
        {
            //  There is nothing to initialize on this
actionhandler.
        }

        /// <summary>
        /// Executes an Action against IBF based on which
menuItem is passed in.
        /// </summary>
        /// <param name="menuItem">The menuItem from the smart
tag that was clicked.</param>
        /// <param name="mediator">IMediator associated with the
host application</param>
        /// <param name="context">The current context.</param>
        public virtual void InvokeMenuItem(
IActionHandlerMenuItem menuItem, IMediator mediator,
IContextInformation context )
        {
            switch ( menuItem.Id.ToString() )
            {
                case
"ShowTimesheets"://Utility.Constants.ShowTimeSheetsMenuItemI
D:
                default:
                    ShowTimeSheets( mediator, context );
                    break;
            }
        }
        #endregion

        #region ShowTimeSheetsMenuItem
        /// <summary>
        /// MenuItem to encapsulate the override Show Details
menu item
        /// </summary>
        private sealed class ShowTimeSheetsMenuItem :
IActionHandlerMenuItem
        {
            private readonly object id;
```

```
            private readonly string caption;

            /// <summary>
            /// Default constructor
            /// </summary>
            public ShowTimeSheetsMenuItem()
            {
                this.caption = "Show Timesheets";
                id = "ShowTimesheets";
            }

            #region IActionHandlerMenuItem Members
            /// <summary>
            /// Gets the object to be used as an identifier for
the menuitem
            /// </summary>
            public object Id
            {
                get { return this.id; }
            }

            /// <summary>
            /// Gets the caption associated with the menuitem
            /// </summary>
            public string Caption
            {
                get { return this.caption; }
            }
            #endregion
        }
        #endregion

    }
}
```

E.2 Constants.cs

The source codes in this file define some constants and static methods, which are used to build the UI regions.

```
using System;
using System.Data;
using System.Globalization;

namespace Maconomy.Sample.UserInterface
{
    /// <summary>
    /// Summary description for Constants.
    /// </summary>
    internal sealed class Constants
    {
        //Variables used to store users' logon info
        internal static string user, code;
        internal static DateTime selectedDate;

        #region IBF Context/Parameter Format Strings
        internal const string TimeSheetContextXmlFormatString =
            "<ContextInformation
MetadataScopeName='Maconomy.SampleSolution.TS'
EntityName='TimeSheet' ViewName='TimeSheetDetails'
ReferenceSchemaName='get (maconomy.com)'
xmlns='http://schemas.microsoft.com/InformationBridge/2004/C
ontextInformation'>"+
            "<Reference>"+
            "<get xmlns='maconomy.com'>"+
            "<refer xmlns=''>"+
            "<username>{0}</username>"+
            "<password>{1}</password>"+
            "<PeriodStart>{2}</PeriodStart>"+
            "</refer>"+
            "</get>"+
            "</Reference>"+
            "</ContextInformation>";

        internal const string RegisterParametersFormatString =
            "<register xmlns='maconomy.com'>"+
            "<refer xmlns=''>"+
            "<username>{0}</username>"+
            "<password>{1}</password>"+
            "<PeriodStart>{2}</PeriodStart>"+
            "</refer>"+
            "<entry xmlns=''>"+
            "<JobNumber>{3}</JobNumber>"+
            "<ActivityNumber>{4}</ActivityNumber>"+
```

```
            "<TaskName>{5}</TaskName>"+
            "<NumberOfDay1>{6}</NumberOfDay1>"+
            "<NumberOfDay2>{7}</NumberOfDay2>"+
            "<NumberOfDay3>{8}</NumberOfDay3>"+
            "<NumberOfDay4>{9}</NumberOfDay4>"+
            "<NumberOfDay5>{10}</NumberOfDay5>"+
            "<NumberOfDay6>{11}</NumberOfDay6>"+
            "<NumberOfDay7>{12}</NumberOfDay7>"+
            "</entry>"+
            "</register>";

        #endregion

        #region IBF Action and Operation Names
        internal const string ShowAddTimesheetLineActionName =
"Register Timesheet Line";
        internal const string SubmitTimesheetLineActionName =
"Submit Timesheet Line";
        #endregion

        //Internal Helper funtion -- get the Monday of the week
in which a day is selected
        internal static DateTime getMon (DateTime date)
        {
            int span = (((int)date.DayOfWeek) + 6) % 7;
            return (date - TimeSpan.FromDays(span));
        }

        //Internal Helper funtion -- get the week number of a
selected day
        internal static int getWeek (DateTime date)
        {
            //DateTime date = getMon(date);
            CultureInfo cult =
CultureInfo.CreateSpecificCulture("da");
            int weekNo = cult.Calendar.GetWeekOfYear(date,
                cult.DateTimeFormat.CalendarWeekRule,
                cult.DateTimeFormat.FirstDayOfWeek);
            return weekNo;
        }

        private Constants()
        {
        }
```

```
    }
}
```

E.3 LogOn.cs

The file contains the sourece codes for the "LogOn" region discussed in section 4.7.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Globalization;
using System.Windows.Forms;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using Microsoft.InformationBridge.Framework.Interfaces;
using Microsoft.InformationBridge.Framework.UI.Interop;


namespace Maconomy.Sample.UserInterface
{
    /// <summary>
    /// Summary description for LogOn.
    /// </summary>
    public class LogOn : UserControl, IRegion
    {
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox username;
        private System.Windows.Forms.TextBox passwd;
        private System.Windows.Forms.PictureBox pictureBox1;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components =
```

```
null;

        public LogOn()
        {
            // This call is required by the Windows.Forms Form
Designer.
            InitializeComponent();

            // TODO: Add any initialization after the
InitComponent call

        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if( components != null )
                    components.Dispose();
            }
            base.Dispose( disposing );
        }

        #region Component Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(LogOn));
            this.label1 = new System.Windows.Forms.Label();
            this.username = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.passwd = new System.Windows.Forms.TextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.pictureBox1 = new
System.Windows.Forms.PictureBox();
            this.SuspendLayout();
```

```
            //
            // label1
            //
            this.label1.BackColor =
System.Drawing.Color.Transparent;
            this.label1.Location = new System.Drawing.Point(40,
56);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(64, 24);
            this.label1.TabIndex = 0;
            this.label1.Text = "Username :";
            //
            // username
            //
            this.username.BackColor =
System.Drawing.Color.Honeydew;
            this.username.Location = new System.Drawing.Point(104,
56);
            this.username.Name = "username";
            this.username.Size = new System.Drawing.Size(96, 20);
            this.username.TabIndex = 1;
            this.username.Text = "";
            this.username.TextChanged += new
System.EventHandler(this.username_TextChanged);
            //
            // label2
            //
            this.label2.BackColor =
System.Drawing.Color.Transparent;
            this.label2.Location = new System.Drawing.Point(40,
104);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 23);
            this.label2.TabIndex = 2;
            this.label2.Text = "Password :";
            //
            // passwd
            //
            this.passwd.BackColor = System.Drawing.Color.Honeydew;
            this.passwd.Location = new System.Drawing.Point(104,
104);
            this.passwd.Name = "passwd";
            this.passwd.PasswordChar = '*';
            this.passwd.Size = new System.Drawing.Size(96, 20);
```

```
            this.passwd.TabIndex = 3;
            this.passwd.Text = "";
            this.passwd.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.passwd_KeyDown);
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.Beige;
            this.button1.Enabled = false;
            this.button1.Location = new System.Drawing.Point(64,
152);
            this.button1.Name = "button1";
            this.button1.TabIndex = 4;
            this.button1.Text = "&Ok";
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.Beige;
            this.button2.Location = new System.Drawing.Point(152,
152);
            this.button2.Name = "button2";
            this.button2.TabIndex = 5;
            this.button2.Text = "&Cancel";
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Ima
ge")));
            this.pictureBox1.Location = new
System.Drawing.Point(0, 0);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(21,
22);
            this.pictureBox1.TabIndex = 6;
            this.pictureBox1.TabStop = false;
            //
            // LogOn
            //
            this.BackColor =
```

```
System.Drawing.SystemColors.ControlLight;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.Backgroun
dImage")));
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.passwd);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.username);
            this.Controls.Add(this.label1);
            this.Name = "LogOn";
            this.Size = new System.Drawing.Size(256, 208);
            this.ResumeLayout(false);

        }
        #endregion

        private void username_TextChanged(object sender,
System.EventArgs e)
        {
            if (username.Text == "")
                button1.Enabled = false;
            else
                button1.Enabled = true;
        }


        // Click event when a user clicks OK button.  This will
invoke Information
        // Bridge to bring up the timesheet data of the current
week.
        private void button1_Click(object sender,
System.EventArgs e)
        {
            //Preparing the context information for the command
            Constants.user = username.Text;
            Constants.code = passwd.Text;
            Constants.selectedDate = DateTime.Today;
            String currentdate =
Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-d
d");
            string formattedString = string.Format(
                CultureInfo.InvariantCulture,
```

```
                Constants.TimeSheetContextXmlFormatString,
                Constants.user, Constants.code, currentdate);
            IContextInformation contextInformation =

Facade.ContextFactory.DeserializeContextInformation(formatte
dString);

            IEngineCommand command =
regionFrameProxy.Host.Mediator.CreateLocalCommand(
                "MSIBF.UI.ShowContext",
                contextInformation.Reference);
            command.Context = contextInformation;
            command.Execute();
        }

        private void button2_Click(object sender,
System.EventArgs e)
        {
            username.Text="";
            passwd.Text="";
        }

        private void passwd_KeyDown(object sender,
System.Windows.Forms.KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter )
            {
                button1_Click(sender, e);
            }
        }

        #region IRegion Members

        private IRegionFrameProxy regionFrameProxy;
        private FrameType hostType;
        private IVisualStyles visualStyles;


        /// <summary>XmlNode with region's data.</summary>
        public XmlNode Data
        {
            set
            {
                // TODO:  Add LogOn.HostProxy setter implement
```

```
            }
        }

        /// <summary>Proxy to the host of the region.</summary>
        public IRegionFrameProxy HostProxy
        {
            set
            {
                this.regionFrameProxy = value;
            }
        }

        /// <summary>Type of host for the region.</summary>
        public FrameType HostType
        {
            set
            {
                this.hostType = value;
            }
        }

        public IVisualStyles VisualStyle
        {
            set
            {
                // TODO:  Add LogOn.VisualStyle setter
implementation
            }
        }

        #endregion
    }
}
```

E.4 DisplayTimesheet.cs

The file contains the sourece codes for the "DisplayTimesheet" region discussed in section 4.7.

```
using System;
```

```
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Globalization;
using System.Windows.Forms;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using Microsoft.InformationBridge.Framework.Interfaces;
using Microsoft.InformationBridge.Framework.UI.Interop;

namespace Maconomy.Sample.UserInterface
{
    /// <summary>
    /// Summary description for DisplayTimesheet.
    /// </summary>
    public class DisplayTimesheet : UserControl, IRegion
    {
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.DataGrid dataGrid1;
        private Tline[] lines = null;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label emno;
        private System.Windows.Forms.Label emname;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label weekno;
        private System.Windows.Forms.Label year;
        private System.Windows.Forms.Label period;
        private System.Windows.Forms.Button register;
        private System.Windows.Forms.DateTimePicker datePicker;
        private System.Windows.Forms.DataGridTableStyle
dataGridTableStyle1;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn1;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn2;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn3;
        private System.Windows.Forms.DataGridTextBoxColumn
```

```
dataGridTextBoxColumn4;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn5;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn6;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn7;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn8;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn9;
        private System.Windows.Forms.DataGridTextBoxColumn
dataGridTextBoxColumn10;
        private System.Windows.Forms.Button submit;
        private System.Windows.Forms.Label approved;
        private System.Windows.Forms.CheckBox approve;
        private System.Windows.Forms.Button show;
        private System.Windows.Forms.GroupBox groupBox3;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components =
null;

        public DisplayTimesheet()
        {
            // This call is required by the Windows.Forms Form
Designer.
            InitializeComponent();

            // TODO: Add any initialization after the
InitializeComponent call

        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
```

```
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Component Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(DisplayTimesheet));
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.emname = new System.Windows.Forms.Label();
            this.emno = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.period = new System.Windows.Forms.Label();
            this.year = new System.Windows.Forms.Label();
            this.weekno = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.dataGrid1 = new System.Windows.Forms.DataGrid();
            this.dataGridTableStyle1 = new
System.Windows.Forms.DataGridTableStyle();
            this.dataGridTextBoxColumn1 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn2 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn3 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn4 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn5 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn6 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn7 = new
```

```
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn8 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn9 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.dataGridTextBoxColumn10 = new
System.Windows.Forms.DataGridTextBoxColumn();
            this.datePicker = new
System.Windows.Forms.DateTimePicker();
            this.register = new System.Windows.Forms.Button();
            this.submit = new System.Windows.Forms.Button();
            this.approved = new System.Windows.Forms.Label();
            this.approve = new System.Windows.Forms.CheckBox();
            this.show = new System.Windows.Forms.Button();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.dataGrid1))
.BeginInit();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.BackColor =
System.Drawing.SystemColors.ActiveBorder;
            this.groupBox1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("groupBox1.Backg
roundImage")));
            this.groupBox1.Controls.Add(this.emname);
            this.groupBox1.Controls.Add(this.emno);
            this.groupBox1.Controls.Add(this.label2);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Font = new
System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.groupBox1.Location = new System.Drawing.Point(8,
16);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(184,
80);
            this.groupBox1.TabIndex = 0;
```

```
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Employee Information";
            //
            // emname
            //
            this.emname.BackColor =
System.Drawing.Color.Transparent;
            this.emname.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.emname.Location = new System.Drawing.Point(52,
48);
            this.emname.Name = "emname";
            this.emname.Size = new System.Drawing.Size(124, 16);
            this.emname.TabIndex = 4;
            //
            // emno
            //
            this.emno.BackColor =
System.Drawing.Color.Transparent;
            this.emno.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.emno.Location = new System.Drawing.Point(80, 24);
            this.emno.Name = "emno";
            this.emno.Size = new System.Drawing.Size(80, 16);
            this.emno.TabIndex = 3;
            //
            // label2
            //
            this.label2.BackColor =
System.Drawing.Color.Transparent;
            this.label2.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label2.Location = new System.Drawing.Point(0,
48);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(48, 16);
            this.label2.TabIndex = 2;
            this.label2.Text = "Name :";
            //
            // label1
            //
```

```
            this.label1.BackColor =
System.Drawing.Color.Transparent;
            this.label1.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label1.Location = new System.Drawing.Point(0,
24);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(80, 16);
            this.label1.TabIndex = 1;
            this.label1.Text = "Employee No.";
            //
            // groupBox2
            //
            this.groupBox2.BackColor =
System.Drawing.SystemColors.ActiveBorder;
            this.groupBox2.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("groupBox2.Backg
roundImage")));
            this.groupBox2.Controls.Add(this.period);
            this.groupBox2.Controls.Add(this.year);
            this.groupBox2.Controls.Add(this.weekno);
            this.groupBox2.Controls.Add(this.label7);
            this.groupBox2.Controls.Add(this.label6);
            this.groupBox2.Controls.Add(this.label5);
            this.groupBox2.Font = new
System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.groupBox2.Location = new
System.Drawing.Point(200, 16);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(208,
80);
            this.groupBox2.TabIndex = 1;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Period";
            //
            // period
            //
            this.period.BackColor =
System.Drawing.Color.Transparent;
            this.period.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
```

```
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.period.Location = new System.Drawing.Point(48,
56);
            this.period.Name = "period";
            this.period.Size = new System.Drawing.Size(152, 16);
            this.period.TabIndex = 10;
            //
            // year
            //
            this.year.BackColor =
System.Drawing.Color.Transparent;
            this.year.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.year.Location = new System.Drawing.Point(72, 32);
            this.year.Name = "year";
            this.year.Size = new System.Drawing.Size(48, 16);
            this.year.TabIndex = 9;
            //
            // weekno
            //
            this.weekno.BackColor =
System.Drawing.Color.Transparent;
            this.weekno.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.weekno.Location = new System.Drawing.Point(72,
16);
            this.weekno.Name = "weekno";
            this.weekno.Size = new System.Drawing.Size(32, 16);
            this.weekno.TabIndex = 8;
            //
            // label7
            //
            this.label7.BackColor =
System.Drawing.Color.Transparent;
            this.label7.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label7.Location = new System.Drawing.Point(8,
56);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(48, 16);
            this.label7.TabIndex = 7;
```

```
            this.label7.Text = "Date :";
            //
            // label6
            //
            this.label6.BackColor =
System.Drawing.Color.Transparent;
            this.label6.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label6.Location = new System.Drawing.Point(8,
32);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(48, 16);
            this.label6.TabIndex = 6;
            this.label6.Text = "Year :";
            //
            // label5
            //
            this.label5.BackColor =
System.Drawing.Color.Transparent;
            this.label5.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label5.Location = new System.Drawing.Point(8,
16);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(48, 16);
            this.label5.TabIndex = 5;
            this.label5.Text = "Week :";
            //
            // dataGrid1
            //
            this.dataGrid1.CaptionBackColor =
System.Drawing.Color.Honeydew;
            this.dataGrid1.CaptionForeColor =
System.Drawing.SystemColors.ControlText;
            this.dataGrid1.CaptionText = "Time Sheet Lines";
            this.dataGrid1.DataMember = "";
            this.dataGrid1.HeaderForeColor =
System.Drawing.SystemColors.ControlText;
            this.dataGrid1.Location = new System.Drawing.Point(8,
160);
            this.dataGrid1.Name = "dataGrid1";
            this.dataGrid1.Size = new System.Drawing.Size(400,
```

```
136);
            this.dataGrid1.TabIndex = 2;
            this.dataGrid1.TableStyles.AddRange(new
System.Windows.Forms.DataGridTableStyle[] {

this.dataGridTableStyle1});
            //
            // dataGridTableStyle1
            //
            this.dataGridTableStyle1.DataGrid = this.dataGrid1;
            this.dataGridTableStyle1.GridColumnStyles.AddRange(new
System.Windows.Forms.DataGridColumnStyle[] {
this.dataGridTextBoxColumn1,
this.dataGridTextBoxColumn2,
this.dataGridTextBoxColumn3,
this.dataGridTextBoxColumn4,
this.dataGridTextBoxColumn5,
this.dataGridTextBoxColumn6,
this.dataGridTextBoxColumn7,
this.dataGridTextBoxColumn8,
this.dataGridTextBoxColumn9,
this.dataGridTextBoxColumn10});

            this.dataGridTableStyle1.HeaderForeColor =
System.Drawing.SystemColors.ControlText;
            this.dataGridTableStyle1.MappingName = "";
            //
            // dataGridTextBoxColumn1
            //
            this.dataGridTextBoxColumn1.Format = "";
            this.dataGridTextBoxColumn1.FormatInfo = null;
            this.dataGridTextBoxColumn1.HeaderText = "Project
No.";
            this.dataGridTextBoxColumn1.MappingName = "ProjectNo";
            this.dataGridTextBoxColumn1.ReadOnly = true;
            this.dataGridTextBoxColumn1.Width = 75;
            //
            // dataGridTextBoxColumn2
            //
            this.dataGridTextBoxColumn2.Format = "";
            this.dataGridTextBoxColumn2.FormatInfo = null;
            this.dataGridTextBoxColumn2.HeaderText = "Act No.";
            this.dataGridTextBoxColumn2.MappingName = "ActNO";
            this.dataGridTextBoxColumn2.ReadOnly = true;
```

```
this.dataGridTextBoxColumn2.Width = 30;
//
// dataGridTextBoxColumn3
//
this.dataGridTextBoxColumn3.Format = "";
this.dataGridTextBoxColumn3.FormatInfo = null;
this.dataGridTextBoxColumn3.HeaderText = "Task";
this.dataGridTextBoxColumn3.MappingName = "Task";
this.dataGridTextBoxColumn3.ReadOnly = true;
this.dataGridTextBoxColumn3.Width = 30;
//
// dataGridTextBoxColumn4
//
this.dataGridTextBoxColumn4.Format = "";
this.dataGridTextBoxColumn4.FormatInfo = null;
this.dataGridTextBoxColumn4.HeaderText = "Monday";
this.dataGridTextBoxColumn4.MappingName = "Monday";
this.dataGridTextBoxColumn4.ReadOnly = true;
this.dataGridTextBoxColumn4.Width = 40;
//
// dataGridTextBoxColumn5
//
this.dataGridTextBoxColumn5.Format = "";
this.dataGridTextBoxColumn5.FormatInfo = null;
this.dataGridTextBoxColumn5.HeaderText = "Tuesday";
this.dataGridTextBoxColumn5.MappingName = "Tuesday";
this.dataGridTextBoxColumn5.ReadOnly = true;
this.dataGridTextBoxColumn5.Width = 40;
//
// dataGridTextBoxColumn6
//
this.dataGridTextBoxColumn6.Format = "";
this.dataGridTextBoxColumn6.FormatInfo = null;
this.dataGridTextBoxColumn6.HeaderText = "Wednesday";
this.dataGridTextBoxColumn6.MappingName = "Wednesday";
this.dataGridTextBoxColumn6.ReadOnly = true;
this.dataGridTextBoxColumn6.Width = 40;
//
// dataGridTextBoxColumn7
//
this.dataGridTextBoxColumn7.Format = "";
this.dataGridTextBoxColumn7.FormatInfo = null;
this.dataGridTextBoxColumn7.HeaderText = "Thursday";
this.dataGridTextBoxColumn7.MappingName = "Thursday";
```

```
            this.dataGridTextBoxColumn7.ReadOnly = true;
            this.dataGridTextBoxColumn7.Width = 40;
            //
            // dataGridTextBoxColumn8
            //
            this.dataGridTextBoxColumn8.Format = "";
            this.dataGridTextBoxColumn8.FormatInfo = null;
            this.dataGridTextBoxColumn8.HeaderText = "Friday";
            this.dataGridTextBoxColumn8.MappingName = "Friday";
            this.dataGridTextBoxColumn8.ReadOnly = true;
            this.dataGridTextBoxColumn8.Width = 40;
            //
            // dataGridTextBoxColumn9
            //
            this.dataGridTextBoxColumn9.Format = "";
            this.dataGridTextBoxColumn9.FormatInfo = null;
            this.dataGridTextBoxColumn9.HeaderText = "Saturday";
            this.dataGridTextBoxColumn9.MappingName = "Saturday";
            this.dataGridTextBoxColumn9.ReadOnly = true;
            this.dataGridTextBoxColumn9.Width = 40;
            //
            // dataGridTextBoxColumn10
            //
            this.dataGridTextBoxColumn10.Format = "";
            this.dataGridTextBoxColumn10.FormatInfo = null;
            this.dataGridTextBoxColumn10.HeaderText = "Sunday";
            this.dataGridTextBoxColumn10.MappingName = "Sunday";
            this.dataGridTextBoxColumn10.ReadOnly = true;
            this.dataGridTextBoxColumn10.Width = 40;
            //
            // datePicker
            //
            this.datePicker.Format =
System.Windows.Forms.DateTimePickerFormat.Short;
            this.datePicker.Location = new
System.Drawing.Point(16, 16);
            this.datePicker.Name = "datePicker";
            this.datePicker.Size = new System.Drawing.Size(120,
20);
            this.datePicker.TabIndex = 3;
            this.datePicker.Value = new System.DateTime(2007, 3,
1, 0, 0, 0, 0);
            this.datePicker.ValueChanged += new
System.EventHandler(this.datePicker_ValueChanged);
```

```
            //
            // register
            //
            this.register.Location = new System.Drawing.Point(40,
304);
            this.register.Name = "register";
            this.register.Size = new System.Drawing.Size(136, 24);
            this.register.TabIndex = 4;
            this.register.Text = "&Add Time Sheet Line";
            this.register.Click += new
System.EventHandler(this.button1_Click);
            //
            // submit
            //
            this.submit.Location = new System.Drawing.Point(208,
304);
            this.submit.Name = "submit";
            this.submit.Size = new System.Drawing.Size(136, 24);
            this.submit.TabIndex = 5;
            this.submit.Text = "&Submit timesheet";
            //
            // approved
            //
            this.approved.BackColor =
System.Drawing.Color.Transparent;
            this.approved.Font = new
System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.approved.Location = new System.Drawing.Point(48,
120);
            this.approved.Name = "approved";
            this.approved.Size = new System.Drawing.Size(64, 16);
            this.approved.TabIndex = 8;
            this.approved.Text = "Approved :";
            //
            // approve
            //
            this.approve.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("approve.Backgro
undImage")));
            this.approve.Enabled = false;
            this.approve.Location = new System.Drawing.Point(112,
120);
```

```
            this.approve.Name = "approve";
            this.approve.Size = new System.Drawing.Size(40, 16);
            this.approve.TabIndex = 9;
            //
            // show
            //
            this.show.Location = new System.Drawing.Point(136,
16);
            this.show.Name = "show";
            this.show.Size = new System.Drawing.Size(56, 20);
            this.show.TabIndex = 11;
            this.show.Text = "&Display";
            this.show.Click += new
System.EventHandler(this.show_Click);
            //
            // groupBox3
            //
            this.groupBox3.BackColor =
System.Drawing.Color.Transparent;
            this.groupBox3.Controls.Add(this.show);
            this.groupBox3.Controls.Add(this.datePicker);
            this.groupBox3.Location = new
System.Drawing.Point(200, 99);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(208,
48);
            this.groupBox3.TabIndex = 12;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Select a timesheet date";
            //
            // DisplayTimesheet
            //
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.Backgroun
dImage")));
            this.Controls.Add(this.groupBox3);
            this.Controls.Add(this.approve);
            this.Controls.Add(this.approved);
            this.Controls.Add(this.submit);
            this.Controls.Add(this.register);
            this.Controls.Add(this.dataGrid1);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Name = "DisplayTimesheet";
```

```
            this.Size = new System.Drawing.Size(416, 336);
            this.Load += new
System.EventHandler(this.DisplayTimesheet_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox2.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.dataGrid1))
.EndInit();
            this.groupBox3.ResumeLayout(false);
            this.ResumeLayout(false);

        }
        #endregion

        // Perfom an IBF action to open the RegisterLine region
        private void button1_Click(object sender,
System.EventArgs e)
        {
            IEngineCommand command =

regionFrameProxy.Host.Mediator.CreateActionByNameCommand(
                Constants.ShowAddTimesheetLineActionName,null);
            command.Context.MetadataScopeName =
"Maconomy.SampleSolution.TS";
            command.Context.EntityName = "TimeSheet";
            command.Context.ViewName = "TimeSheetDetails";

            command.Execute();
        }

        private void datePicker_Enter(object sender,
System.EventArgs e)
        {

            String newdate =
Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-d
d");
            string formattedString = string.Format(
                CultureInfo.InvariantCulture,
                Constants.TimeSheetContextXmlFormatString,
                Constants.user, Constants.code, newdate);

            IContextInformation contextInformation =
Facade.ContextFactory.DeserializeContextInformation(formatte
```

```
dString);

            IEngineCommand command =
regionFrameProxy.Host.Mediator.CreateLocalCommand(
                "MSIBF.UI.ShowContext",
                contextInformation.Reference);
            command.Context = contextInformation;
            command.Execute();

        }

        private void datePicker_ValueChanged(object sender,
System.EventArgs e)
        {
            datePicker.Text =
datePicker.Value.ToString("yyyy-MM-dd");
            //Constants.selectedDate = datePicker.Value;
        }

        private void DisplayTimesheet_Load(object sender,
System.EventArgs e)
        {
            datePicker.Value = Constants.selectedDate;
        }

        private void show_Click(object sender, System.EventArgs
e)
        {
            Constants.selectedDate = datePicker.Value;
            String newdate =
Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-d
d");
            string formattedString = string.Format(
                CultureInfo.InvariantCulture,
                Constants.TimeSheetContextXmlFormatString,
                Constants.user, Constants.code, newdate);

            IContextInformation contextInformation =
Facade.ContextFactory.DeserializeContextInformation(formatte
dString);

            IEngineCommand command =
regionFrameProxy.Host.Mediator.CreateLocalCommand(
                "MSIBF.UI.ShowContext",
```

```
                contextInformation.Reference);
            command.Context = contextInformation;
            command.Execute();
        }

        #region IRegion Members

        private IRegionFrameProxy regionFrameProxy;
        private FrameType hostType;
        private IVisualStyles visualStyles;

        /// <summary>XmlNode with region's data.</summary>
        ///

        public XmlNode Data
        {
            set
            {
                XmlNode dataNode = value;
//Call the LoadXmlData routine to insert data in UI
//controls
                LoadXmlData(dataNode);
            }
        }


        /// <summary>Proxy to the host of the region.</summary>
        ///

        public IRegionFrameProxy HostProxy
        {
            set
            {
                this.regionFrameProxy = value;
            }

        }

        /// <summary>Type of host for the region.</summary>
        public FrameType HostType
        {
            set
            {
                this.hostType = value;
```

```
            }
        }

        public IVisualStyles VisualStyle
        {
            set
            {
                // TODO:  Add LogOn.VisualStyle setter
implementation
            }
        }

        #endregion

        #region Data transfer from region data to user interface
        private void LoadXmlData(XmlNode data)
        {
            if (data["EmployeeNumber"] != null )
            {
                emno.Text = data["EmployeeNumber"].InnerText;
                emname.Text = Constants.user;
            }
            if (data["Approved"] != null )
            {
                if (data["Approved"].InnerText.Equals("true"))
                    approve.Checked = true;
                else
                    approve.Checked = false;
            }

            if (null != data["PeriodStart"])
            {
                DateTime start =
DateTime.ParseExact(data["PeriodStart"].InnerText,
"yyyy-MM-dd", null);
                DateTime end = start + TimeSpan.FromDays(6);
                period.Text = start.ToString("MM/dd/yyyy")
                + " - "+ end.ToString("MM/dd/yyyy");
                year.Text = start.Year.ToString();
                weekno.Text = Constants.getWeek(start).ToString();
            }
            else
            {
                period.Text = "";
```

```
                year.Text = "";
                weekno.Text = "";
            }
            this.ShowLines(data["Rows"]);

        }

/// <summary>
/// Helper function used to display timesheet lines in a
//DataGrid control.
/// </summary>
/// <param name="rows"></param>
    private void ShowLines(XmlElement rows)
    {
    int num = rows.GetElementsByTagName("element").Count;
    if (num > 0)
    {
        lines = new Tline[num];
        XmlNode node = rows.FirstChild;

        for (int i = 0; i < num; i++ )
        {
        lines[i] = new Tline();
        lines[i].ProjectNo = node["JobNumber"].InnerText;
        lines[i].ActNo = node["ActivityNumber"].InnerText;
        lines[i].Task = node["TaskName"].InnerText;
        lines[i].Monday = node["NumberOfDay1"].InnerText;
        lines[i].Tuesday = node["NumberOfDay2"].InnerText;
        lines[i].Wednesday =
    node["NumberOfDay3"].InnerText;
        lines[i].Thursday =
    node["NumberOfDay4"].InnerText;
        lines[i].Friday =
        node["NumberOfDay5"].InnerText;
        lines[i].Saturday =
    node["NumberOfDay6"].InnerText;
        lines[i].Sunday = node["NumberOfDay7"].InnerText;
        node = node.NextSibling;
        }
    }
        dataGrid1.DataSource = lines;
        dataGridTableStyle1.MappingName =
    lines.GetType().Name;
      dataGrid1.TableStyles.Add(dataGridTableStyle1);
```

```
}

        #endregion // Data transfer from region data to user
interface
    }

    public class Tline
    {
        private string projectNo;
        private string actNo;
        private string task;
        private string monday;
        private string tuesday;
        private string wednesday;
        private string thursday;
        private string friday;
        private string saturday;
        private string sunday;

        public Tline()
        {
        }

        public Tline(string ProjectNo, string ActNo, string
Task, string Monday,
            string tues, string weds, string thurs, string fri,
string sat, string Sunday)
        {
            projectNo=ProjectNo;
            actNo=ActNo;
            task=Task;
            monday = Monday;
            tuesday = tues;
            wednesday = weds;
            thursday = thurs;
            friday = fri;
            saturday = sat;
            sunday = Sunday;
        }

        public string ProjectNo
        {
            get
            {
```

```
            return projectNo;
        }
        set
        {
            projectNo = value;
        }
    }

    public string ActNo
    {
        get
        {
            return actNo;
        }
        set
        {
            actNo = value;
        }
    }

    public string Task
    {
        get
        {
            return task;
        }
        set
        {
            task = value;
        }
    }

    public string Monday
    {
        get
        {
            return monday;
        }
        set
        {
            monday = value;
        }
    }
```

```
public string Tuesday
{
    get
    {
        return tuesday;
    }
    set
    {
        tuesday = value;
    }
}

public string Wednesday
{
    get
    {
        return wednesday;
    }
    set
    {
        wednesday = value;
    }
}

public string Thursday
{
    get
    {
        return thursday;
    }
    set
    {
        thursday = value;
    }
}

public string Friday
{
    get
    {
        return friday;
    }
    set
    {
```

```
            friday = value;
        }
    }

    public string Saturday
    {
        get
        {
            return saturday;
        }
        set
        {
            saturday = value;
        }
    }

    public string Sunday
    {
        get
        {
            return sunday;
        }
        set
        {
            sunday = value;
        }
    }
}

}
```

E.5 RegisterLine.cs

The file contains the sourece codes for the "RegisterLine" region discussed in section 4.7.

```
using System;
using System.Globalization;
using System.Collections;
using System.ComponentModel;
```

```
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using Microsoft.InformationBridge.Framework.Interfaces;
using Microsoft.InformationBridge.Framework.UI.Interop;

namespace Maconomy.Sample.UserInterface
{
    /// <summary>
    /// Summary description for RegisterLine.
    /// </summary>
    public class RegisterLine : UserControl, IRegion
    {
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox projectNo;
        private System.Windows.Forms.TextBox actNo;
        private System.Windows.Forms.TextBox task;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.TextBox weds;
        private System.Windows.Forms.TextBox tues;
        private System.Windows.Forms.TextBox monday;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.TextBox sat;
        private System.Windows.Forms.TextBox fri;
        private System.Windows.Forms.TextBox thur;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.TextBox sunday;
        private System.Windows.Forms.Button ok;
        private System.Windows.Forms.Button cancel;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components =
null;
```

```
        private bool errorOccurred = false;

        public RegisterLine()
        {
            // This call is required by the Windows.Forms Form
Designer.
            InitializeComponent();

            // TODO: Add any initialization after the
InitializeComponent call

        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Component Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(RegisterLine));
            this.label1 = new System.Windows.Forms.Label();
            this.projectNo = new System.Windows.Forms.TextBox();
            this.actNo = new System.Windows.Forms.TextBox();
            this.task = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
```

```
            this.label5 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.weds = new System.Windows.Forms.TextBox();
            this.tues = new System.Windows.Forms.TextBox();
            this.monday = new System.Windows.Forms.TextBox();
            this.label8 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.sat = new System.Windows.Forms.TextBox();
            this.fri = new System.Windows.Forms.TextBox();
            this.thur = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.sunday = new System.Windows.Forms.TextBox();
            this.ok = new System.Windows.Forms.Button();
            this.cancel = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.BackColor =
System.Drawing.Color.Transparent;
            this.label1.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label1.Location = new System.Drawing.Point(8, 8);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(240, 24);
            this.label1.TabIndex = 0;
            this.label1.Text = "Register a timesheet line by
filling the form :";
            //
            // projectNo
            //
            this.projectNo.Location = new System.Drawing.Point(8,
56);
            this.projectNo.Name = "projectNo";
            this.projectNo.Size = new System.Drawing.Size(64, 20);
            this.projectNo.TabIndex = 1;
            this.projectNo.Text = "2310001";
            //
            // actNo
            //
            this.actNo.Location = new System.Drawing.Point(72,
```

```
56);
            this.actNo.Name = "actNo";
            this.actNo.Size = new System.Drawing.Size(56, 20);
            this.actNo.TabIndex = 2;
            this.actNo.Text = "120";
            //
            // task
            //
            this.task.Location = new System.Drawing.Point(128,
56);
            this.task.Name = "task";
            this.task.Size = new System.Drawing.Size(56, 20);
            this.task.TabIndex = 3;
            this.task.Text = "240";
            //
            // label2
            //
            this.label2.AllowDrop = true;
            this.label2.BackColor =
System.Drawing.Color.Gainsboro;
            this.label2.Location = new System.Drawing.Point(8,
40);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 16);
            this.label2.TabIndex = 4;
            this.label2.Text = "Project No.";
            //
            // label3
            //
            this.label3.AllowDrop = true;
            this.label3.BackColor =
System.Drawing.Color.Gainsboro;
            this.label3.Location = new System.Drawing.Point(72,
40);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(56, 16);
            this.label3.TabIndex = 5;
            this.label3.Text = "Act.  No.";
            //
            // label4
            //
            this.label4.AllowDrop = true;
            this.label4.BackColor =
System.Drawing.Color.Gainsboro;
```

```
            this.label4.Location = new System.Drawing.Point(128,
40);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(56, 16);
            this.label4.TabIndex = 6;
            this.label4.Text = "Task";
            //
            // label5
            //
            this.label5.AllowDrop = true;
            this.label5.BackColor =
System.Drawing.Color.Gainsboro;
            this.label5.Location = new System.Drawing.Point(128,
88);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(64, 16);
            this.label5.TabIndex = 12;
            this.label5.Text = "Wednesday";
            //
            // label6
            //
            this.label6.AllowDrop = true;
            this.label6.BackColor =
System.Drawing.Color.Gainsboro;
            this.label6.Location = new System.Drawing.Point(72,
88);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(56, 16);
            this.label6.TabIndex = 11;
            this.label6.Text = "Tuesday";
            //
            // label7
            //
            this.label7.AllowDrop = true;
            this.label7.BackColor =
System.Drawing.Color.Gainsboro;
            this.label7.Location = new System.Drawing.Point(8,
88);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(64, 16);
            this.label7.TabIndex = 10;
            this.label7.Text = "Monday";
            //
            // weds
```

```
            //
            this.weds.Location = new System.Drawing.Point(128,
104);
            this.weds.Name = "weds";
            this.weds.Size = new System.Drawing.Size(64, 20);
            this.weds.TabIndex = 9;
            this.weds.Text = "0.0";
            //
            // tues
            //
            this.tues.Location = new System.Drawing.Point(72,
104);
            this.tues.Name = "tues";
            this.tues.Size = new System.Drawing.Size(56, 20);
            this.tues.TabIndex = 8;
            this.tues.Text = "0.0";
            //
            // monday
            //
            this.monday.Location = new System.Drawing.Point(8,
104);
            this.monday.Name = "monday";
            this.monday.Size = new System.Drawing.Size(64, 20);
            this.monday.TabIndex = 7;
            this.monday.Text = "0.0";
            //
            // label8
            //
            this.label8.AllowDrop = true;
            this.label8.BackColor =
System.Drawing.Color.Gainsboro;
            this.label8.Location = new System.Drawing.Point(312,
88);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(56, 16);
            this.label8.TabIndex = 18;
            this.label8.Text = "Saturday";
            //
            // label9
            //
            this.label9.AllowDrop = true;
            this.label9.BackColor =
System.Drawing.Color.Gainsboro;
            this.label9.Location = new System.Drawing.Point(256,
```

```
88);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(56, 16);
            this.label9.TabIndex = 17;
            this.label9.Text = "Friday";
            //
            // label10
            //
            this.label10.AllowDrop = true;
            this.label10.BackColor =
System.Drawing.Color.Gainsboro;
            this.label10.Location = new System.Drawing.Point(192,
88);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(64, 16);
            this.label10.TabIndex = 16;
            this.label10.Text = "Thursday";
            //
            // sat
            //
            this.sat.Location = new System.Drawing.Point(312,
104);
            this.sat.Name = "sat";
            this.sat.Size = new System.Drawing.Size(56, 20);
            this.sat.TabIndex = 15;
            this.sat.Text = "0.0";
            //
            // fri
            //
            this.fri.Location = new System.Drawing.Point(256,
104);
            this.fri.Name = "fri";
            this.fri.Size = new System.Drawing.Size(56, 20);
            this.fri.TabIndex = 14;
            this.fri.Text = "0.0";
            //
            // thur
            //
            this.thur.Location = new System.Drawing.Point(192,
104);
            this.thur.Name = "thur";
            this.thur.Size = new System.Drawing.Size(64, 20);
            this.thur.TabIndex = 13;
            this.thur.Text = "0.0";
```

```
            //
            // label11
            //
            this.label11.AllowDrop = true;
            this.label11.BackColor =
System.Drawing.Color.Gainsboro;
            this.label11.Location = new System.Drawing.Point(368,
88);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(56, 16);
            this.label11.TabIndex = 20;
            this.label11.Text = "Sunday";
            //
            // sunday
            //
            this.sunday.Location = new System.Drawing.Point(368,
104);
            this.sunday.Name = "sunday";
            this.sunday.Size = new System.Drawing.Size(56, 20);
            this.sunday.TabIndex = 19;
            this.sunday.Text = "0.0";
            //
            // ok
            //
            this.ok.Location = new System.Drawing.Point(88, 136);
            this.ok.Name = "ok";
            this.ok.Size = new System.Drawing.Size(72, 24);
            this.ok.TabIndex = 21;
            this.ok.Text = "&Submit";
            this.ok.Click += new
System.EventHandler(this.ok_Click);
            //
            // cancel
            //
            this.cancel.Location = new System.Drawing.Point(248,
136);
            this.cancel.Name = "cancel";
            this.cancel.Size = new System.Drawing.Size(72, 24);
            this.cancel.TabIndex = 22;
            this.cancel.Text = "&Cancel";
            this.cancel.Click += new
System.EventHandler(this.cancel_Click);
            //
            // RegisterLine
```

```
            //
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.Backgroun
dImage")));
            this.Controls.Add(this.cancel);
            this.Controls.Add(this.ok);
            this.Controls.Add(this.label11);
            this.Controls.Add(this.sunday);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.sat);
            this.Controls.Add(this.fri);
            this.Controls.Add(this.thur);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.weds);
            this.Controls.Add(this.tues);
            this.Controls.Add(this.monday);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.task);
            this.Controls.Add(this.actNo);
            this.Controls.Add(this.projectNo);
            this.Controls.Add(this.label1);
            this.Name = "RegisterLine";
            this.Size = new System.Drawing.Size(432, 176);
            this.ResumeLayout(false);

        }
        #endregion

//Click event when a user clicks 'Submit'.  This will
invoke Information Bridge
//to add a timesheet line into the current timesheet
with values the user input.
        private void ok_Click(object sender, System.EventArgs e)
        {
//Construct the parameter object for the command
//to be created
            string date =
Constants.getMon(Constants.selectedDate).ToString("yyyy-MM-d
```

```
d");
            string actionParameters = String.Format(
                CultureInfo.InvariantCulture,
                Constants.RegisterParametersFormatString,
                Constants.user, Constants.code, date,
                projectNo.Text, actNo.Text, task.Text,
                monday.Text, tues.Text, weds.Text,
                thur.Text, fri.Text, sat.Text, sunday.Text);

            IEngineCommand command =

   regionFrameProxy.Host.Mediator.CreateActionByNameCommand(
                Constants.SubmitTimesheetLineActionName,null);
            command.Context.MetadataScopeName =
  "Maconomy.SampleSolution.TS";
    command.Context.EntityName = "TimeSheet";
    command.Context.ViewName = "TimeSheetDetails";
    command.Context.Parameters = actionParameters;

    //For error handling
    errorOccurred = false;
    command.Error += new
    ErrorEventHandler(command_Error);
    command.Execution += new
  ExecutionEventHandler(command_Execution);
            command.Execute();
        }

        private void cancel_Click(object sender,
System.EventArgs e)
        {
            ParentForm.Close();
        }

        private void command_Error(object sender, ErrorEventArgs
e)
        {
            errorOccurred = true;

            MessageBox.Show(
"Fail to register the current timesheet line,
Please try again!", "Submission Error");
        }
```

```
        /// <summary>
        /// Catch the execution event and process any
information that we need to do.
        /// </summary>
        /// <param name="sender">The object which fired the
event</param>
        /// <param name="e">The event Arguments</param>
        private void command_Execution(object sender,
ExecutionEventArgs e)
        {
            if ((null != e) &&
                (null != e.Command) &&
                (CommandStatus.Finished == e.Command.Status))
            {
                if (errorOccurred)
                {
                    Enabled = true;
                }
                else
                {
                    ParentForm.Close();
                }
            }
        }

        #region IRegion Members

        private IRegionFrameProxy regionFrameProxy;
        private FrameType hostType;
        private IVisualStyles visualStyles;



        /// <summary>XmlNode with region's data.</summary>
        public XmlNode Data
        {
            set
            {
                // TODO:  Add RegisterLine.HostProxy setter
implement
            }
        }

        /// <summary>Proxy to the host of the region.</summary>
```

```
        public IRegionFrameProxy HostProxy
        {
            set
            {
                this.regionFrameProxy = value;
            }
        }

        /// <summary>Type of host for the region.</summary>
        public FrameType HostType
        {
            set
            {
                this.hostType = value;
            }
        }

        public IVisualStyles VisualStyle
        {
            set
            {
                // TODO:  Add RegisterLine.VisualStyle setter
implementation
            }
        }

        #endregion

    }
}
```