

**Project number: IST-2001-52222**



*Acceleration of Innovative Ideas to Market*

### **Deliverable 3.3**

## **WEB Infrastructure Specification**

### **Consortium Partners:**

LABEIN	E
Ball Packaging Europe	D
Charles Robinson (Cutting Tools)	UK
MB Air Systems Ltd.	UK
ATOS Origin	E
ATB	D
iCIMS I	CH

Document Ref.: AIM/WP3/D3.3

Version: 1.0

Date: 31/08/2006

Distribution: Public

---

This document is a property of the AIM consortium. You may not copy or use it in part without written permission.

## Executive Summary

The purpose of this document is to define the specification for the development of Web Services to realise an interface between the already-developed software system for Project AIM with the external world. The objective of this writing is that of defining and describing a *standard interface* that allows the use of the AIM system within *legacy systems* and *on the Word Wide Web*.

## DOCUMENT CHANGE LOG

### Change Control Procedures

Each change or set of changes made to this document will result in an increment to the version number of the document. Minor changes will increment the decimal point of the version number. The change log will record this process and will identify for each version number of the document any modification(s), which caused the version number to be raised.

Version	Date	Reason for Change
1.0	31/08/2004	Original Document Creation

---

**TABLE OF CONTENTS**

<b>Introduction and Context Delimitation .....</b>	<b>7</b>
1.1 Project Scope .....	7
1.2 Interdependency.....	7
1.3 Goals.....	7
<b>2 XTra.NET Interface WebService .....</b>	<b>9</b>
2.1 Use of the System .....	9
2.2 Diagram showing the use of the system .....	10
2.3 List Of the Methods To Implement .....	11
2.3.1 Purpose of this section.....	11
2.3.2 Login .....	11
2.3.3 Change Password .....	12
2.3.4 GetMenu.....	14
2.3.5 GetPage .....	15
2.3.6 Description .....	15
2.3.7 GetFile.....	18
2.4 Exception Handling .....	20
2.5 Authentication handling .....	21
2.6 Diagram of the Authentication Classes .....	22
2.7 Diagram of the Classes .....	23
<b>3 Web Service Interface for Legacy Systems.....</b>	<b>24</b>
3.1 List of the Methods.....	24
3.1.1 Read.....	24
3.1.2 ReadDecoded.....	24
3.1.3 List.....	24
3.1.4 ListForUpdate.....	24
3.1.5 Save .....	24
3.2 Exception Handling .....	25
3.3 Authentication handling .....	25
<b>4 Data Structures.....</b>	<b>25</b>
4.1 ExceptionTypeEnum .....	25
4.2 ExceptionStructure .....	25
4.2.1 Example.....	26
4.3 DataStructure .....	26
4.3.1 Example.....	26
4.4 Page.....	27
4.4.1 Example 1 - Documents Browser.....	27
4.4.2 Example 2 - Links Edit.....	301
4.4.3 Example 3 – Documents Edit.....	32
4.5 PageItem .....	334
4.5.1 Class Diagram .....	34

---

4.6	TextSection	34
4.6.1	Example	34
4.7	GridSection	34
4.7.1	Example	35
4.8	ButtonSection	35
4.8.1	Example	35
4.9	Button	35
4.9.1	Example	35
4.10	LinkSection	36
4.10.1	Example	36
4.11	Link	36
4.11.1	Example	36
4.12	NavigationTypeEnum	37
4.13	OrientationEnum	37
4.14	Form	37
4.14.1	Example	37
4.15	FormItem	37
4.16	HiddenField	38
4.16.1	Example	38
4.17	TextBox	38
4.17.1	Example	39
4.18	TextBoxString	39
4.18.1	Example	39
4.19	TextBoxNumber	39
4.19.1	Example	39
4.20	InputDate	39
4.20.1	Example	40
4.21	InputFile	40
4.21.1	Example	40
4.22	Option	41
4.23	MultipleChoice	41
4.24	DropDownList	41
4.24.1	Example	41
4.25	RadioButtons	42
4.25.1	Example	42
4.26	TreeView	42
4.26.1	Example	42
4.27	TreeNode	42
4.27.1	Example	42
4.28	CheckBox	43
4.28.1	Example	43
4.29	Menu	43
4.29.1	Example	43
4.30	MenuItem	44
4.30.1	Example	44

---

**5 Conclusion.....44**

---

**ABBREVIATIONS**

ATB	Institute für angewandte Systemtechnik Bremen GmbH
BPE	Ball Packaging Europe
CBR	Case-Based Reasoning
CK	Corporate Knowledge
C-Tools	Cutting Tools
e.g.	"exempli gratia" = engl. For example
GUI	Graphical User Interface
i.e.	"id est" = engl. That is
iCIMS I	Institute CIMS I of SUPSI, Manno, CH
IT	Information Technology
IST	Information Society Technology
LTV	Logovisual Technology
MBAS	MB Air Systems
OO	Object Oriented
PI	Process improvements
PM	Project Manager
QC	Quality Control
RBR	Rule-Based Reasoning
R&D	Research and Development
SE	Software Engineering
ATOS	ATOS Origin
SME	Small and Medium sized Enterprises
SRS	System Requirements Specifications
SUPSI	Scuola Universitaria Professionale della Svizzera Italiana, Manno, CH
t.b.d.	to be defined
TQM	Total Quality Management
TR	Temporal Reasoning
w.r.t.	With respect to
WP	Workpackage
Xtra.NET	The Web system developed at iCIMS I

## Introduction and Context Delimitation

This section introduces the work performed and identifies the boundaries of the context, in order to clarify the aim of the realisation of the web infrastructure that interfaces the AIM System to the World Wide Web.

XTra.NET is a software system developed at the institute iCIMSI of the SUPSI, the University for Applied Science of Southern Switzerland. For further details, see

*S.Capuano, S.Aquilini, D.Baggi, A System For Managing Information in Concurrent Engineering*, Proceedings of the 10<sup>th</sup> ISPE International Conference on Concurrent Engineering: Research and Applications, 26-30 July 2003, Madeira, Portugal, pp.1151-1158.

### 1.1 Project Scope

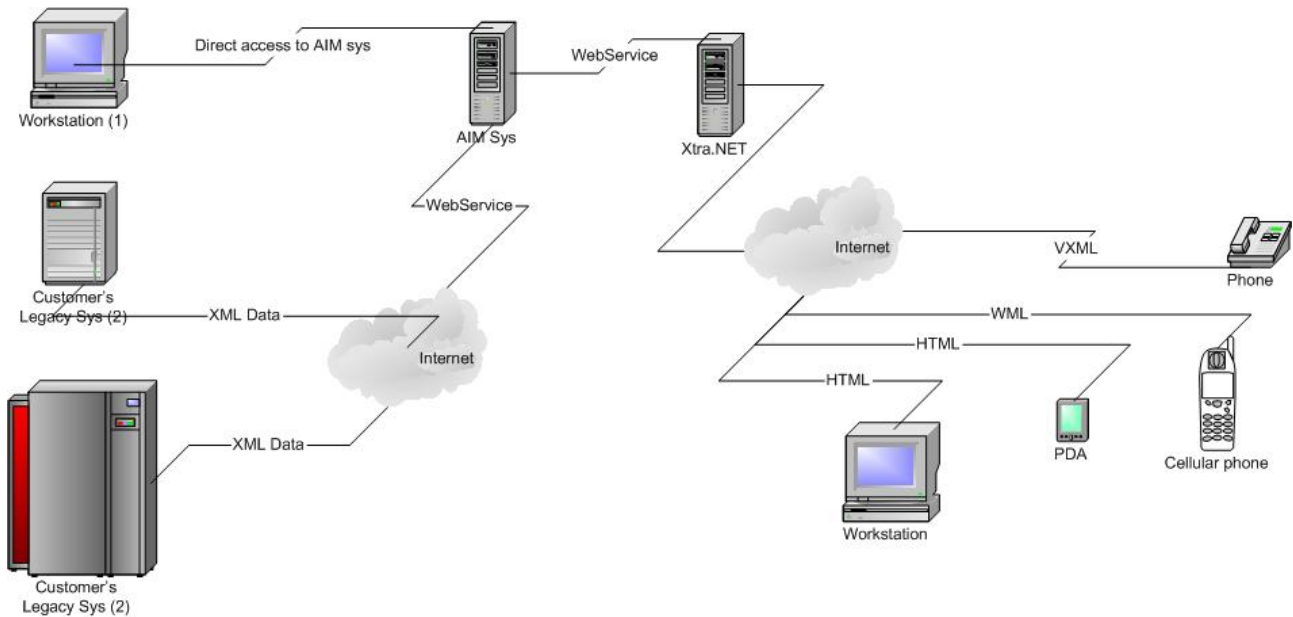
The corresponding Work Package aims, on one hand, at developing a new presentation layer that allows interaction with the AIM System through the Web and, on the other hand, at opening the system to future integration, in particular within existing legacy systems, and extensions.

### 1.2 Interdependency

This solution has been chosen in order to reduce any possible impact on the existing system and to minimise any interdependency among the existing development teams and the various software components that have to be integrated. This approach guarantees the maximum flexibility in regard to possible requirement changes during project development, and the greatest independence of the various teams responsible for development within the whole project. The architecture proposed in this document guarantees any possible evolution of the entire system and the later addition of Web interfaces, without the need to modify the Web Presentation Layer described below.

### 1.3 Goals

The main goal consists of developing a standard interface for the modules of the AIM system, in order to provide a standardised way to access the AIM Application Service. The picture and description below are used to show this claim.



Standardisation of access to a software system such as AIM is achieved, we claim, thanks to the implementation of a set of WebServices. This means that the WebServices will be accessed, on one hand, through a Web system like our *XTra.NET*, which provides a Web interface; and, on the other hand, from a customer's legacy systems. Hence, the whole allows integration of new AIM functionality into existing systems.

The interesting feature of this concept is that the AIM system can then be considered a service, and not an application. If one wants, he/she can still access AIM by using a workstation directly connected to it – even though that requires installation of a complete infrastructure (servers, software, system manager, etc), something that can be afforded only by a big company. However, a system will have more appeal on the market if customers' financial situations are taken into account, and the best way to do so is to provide a *service*, instead of a system.

In such a case, the customer will be able to interact with the AIM system through *XTra.NET* only with a simple web browser as Web interface, which acts as a new presentation layer. The user will therefore buy a *service* and not, as in the past, an *application*. At the same time, the customer also has the opportunity to integrate the AIM functionality into a *legacy system*, by directly accessing the web services made available by the AIM system. This services consist of a set of "Standard API", provided as standard WebServices (SOAP/XML). In this case, the customer pays only for the use of the system – for example, a fee per call – and is not charged for the cost of the infrastructure for the entire system. As a result, SME's with a limited budget will be able to access the service, because no considerable investment is needed.

## 2 XTra.NET Interface WebService

### 2.1 Use of the System

The whole system has been designed with considerable built-in flexibility in respect to possible, later, modifications. The basic concept of interfacing to the Web has been that of analysing what the AIM system visualises, and how it does so. This means that structures of the Web pages are supplied by AIM to Xtra.NET *at runtime*, implying that such structures, including the shape of the pages and the like, can vary at any moment, without the need to modify anything in the Xtra.Net code developed. Thus, maximal independence of the system is achieved in regard to its representation.

This section is a description in words of such features. For a better understanding, reference is made to the diagram of the following section, which described the conceptual functioning of the solution proposed.

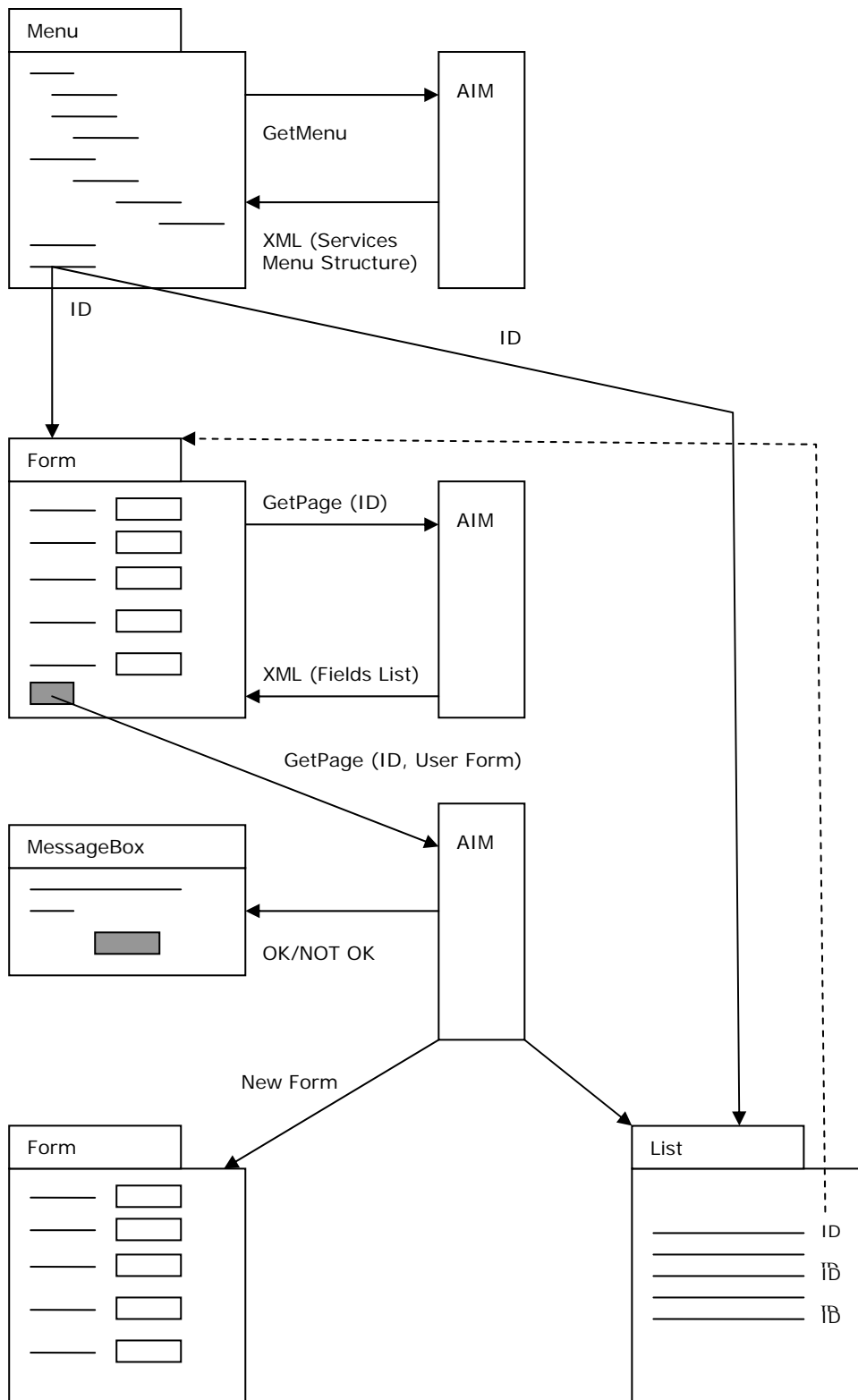
In order to visualise a list of functions that has be displayed, e.g. a menu, XTra.NET issues a call to the particular Webs server – of which there can be more than one. Recall that there are three development teams in this project, each of which is developing, or has developed, three different services for the Web. Hence, the system will access every particular Web service, correctly issuing a call to each: this is method *GetMenu*. The calls return a list of the functions to display, the available *forms* and assign to each a unique identifier.

Thereafter, upon selection by the user of a menu item, XTra.NET calls the Web service, method *GetPage*, on the server which has exposed the functions as menu items, passing to it the received identifier. The called Web service, depending on the identifier, will return to XTra.NET an object that describes the page to display. Thanks to this simple structure, it is possible to manage practically every situation.

As an example, consider the time sequence of a screen for searching and editing:

1. *Form* to insert the information
2. Click on button “search”, which sends the *form* with the data inserted by the user to the Web service
3. The Web service returns again the form with the selected data and, as result, a table with the records found
4. If the user wants to modify the selection criteria, the process restarts at point 1
5. The user selects a record
6. The Web service gets called with the identifier of the selected record
7. The Web service returns the details of the selected record as a *Form* object
8. Upon clicking a button such as *delete*, *update*, etc., the Web service gets called with the selected event and the *data form*
9. The Web service analyses the data and executes the selected operation on the data
10. The Web service returns a page with a message to the user about the state of the operation.

## 2.2 Diagram showing the use of the system



## 2.3 List Of the Methods To Implement

### 2.3.1 Purpose of this section

The specification of the methods that are used by XTra.Net for the Web interface of AIM services is given herewith.

### 2.3.2 Login

#### 2.3.2.1 Description

This method verifies the credential of the user at *login time*. Its input parameters are *UserName* and *Password* and returns an *AuthenticationData* object, which contains *UserID*, *GroupID*, etc., provided if the login is successful. More details are given in Section “2.5 Authentication handling” on page 21.

#### 2.3.2.2 SOAP

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

##### REQUEST:

```
POST /Wsaim/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.icimsi.ch/Login"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Login xmlns="http://www.icimsi.ch">
      <UserName>string</UserName>
      <Password>string</Password>
    </Login>
  </soap:Body>
</soap:Envelope>
```

##### RESPONSE:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```

    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LoginResponse xmlns="http://www.icimsi.ch">
      <LoginResult>
        <UserData>
          <Logged>boolean</Logged>
          <UserID>string</UserID>
          <UserName>string</UserName>
          <Password>string</Password>
          <GroupID>string</GroupID>
          <AdditionalInformation>
            <Parameter xsi:nil="true" />
            <Parameter xsi:nil="true" />
          </AdditionalInformation>
        </UserData>
        <ForceChangePassword>boolean</ForceChangePassword>
        <Exception>
          <Message>string</Message>
          <Description>string</Description>
          <ExceptionType>SystemException or LogicalException or
            SystemWarning or LogicalWarning</ExceptionType>
        </Exception>
      </LoginResult>
    </LoginResponse>
  </soap:Body>
</soap:Envelope>

```

### 2.3.3 Change Password

#### 2.3.3.1 Description

This method allows the user to change his/her password. Its input parameters are *User Name*, *Old Password* and *New Password* and it returns structure *AuthenticationData*, which contains *UserID*, *GroupID*, etc. with the new data of the user. More details are given in Section “2.5 Authentication handling” on page 21.

This method is practically identical to *login* and must perform authentication, but in addition it changes the user’s password.

#### 2.3.3.2 SOAP

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

```

POST /Wsaim/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.icimsi.ch/ChangePassword"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

```

```

    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <ChangePassword xmlns="http://www.icimsi.ch">
    <UserName>string</UserName>
    <OldPassword>string</OldPassword>
    <NewPassword>string</NewPassword>
  </ChangePassword>
</soap:Body>
</soap:Envelope>

```

**RESPONSE:**

HTTP/1.1 200 OK  
 Content-Type: text/xml; charset=utf-8  
 Content-Length: **length**

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ChangePasswordResponse xmlns="http://www.icimsi.ch">
      <LoginResult>
        <UserData>
          <Logged>boolean</Logged>
          <UserID>string</UserID>
          <UserName>string</UserName>
          <Password>string</Password>
          <GroupID>string</GroupID>
          <AdditionalInformation>
            <Parameter xsi:nil="true" />
            <Parameter xsi:nil="true" />
          </AdditionalInformation>
        </UserData>
        <ForceChangePassword>boolean</ForceChangePassword>
        <Exception>
          <Message>string</Message>
          <Description>string</Description>
          <ExceptionType>SystemException or LogicalException or
            SystemWarning or LogicalWarning</ExceptionType>
        </Exception>
      </LoginResult>
    </ChangePasswordResponse>
  </soap:Body>
</soap:Envelope>

```

## 2.3.4 GetMenu

### 2.3.4.1 Description

This method returns a data structure *Menu* that contains a hierarchical list of elements of type *MenuItem*. Details are given in Section “4 Data Structures” on page 25. The call passes an object *AuthenticationData* that contains information about the user currently logged in, so that the method can determine, in function of the user permission, which menu items are to be returned or not.

### 2.3.4.2 SOAP

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

#### REQUEST:

```
POST /Wsaim/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.icimsi.ch/GetMenu"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetMenu xmlns="http://www.icimsi.ch">
      <AuthenticationData>
        <Logged>boolean</Logged>
        <UserID>string</UserID>
        <UserName>string</UserName>
        <Password>string</Password>
        <GroupID>string</GroupID>
        <AdditionalInformation>
          <Parameter>
            <Name>string</Name>
            <Value>string</Value>
          </Parameter>
          <Parameter>
            <Name>string</Name>
            <Value>string</Value>
          </Parameter>
        </AdditionalInformation>
      </AuthenticationData>
    </GetMenu>
  </soap:Body>
</soap:Envelope>
```

#### RESPONSE:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetMenuResponse xmlns="http://www.icimsi.ch">
      <Menu>
        <MenuItems>
          <MenuItem>
            <Label>string</Label>
            <Description>string</Description>
            <NavigationPageID>string</NavigationPageID>
            <Childs xsi:nil="true" />
          </MenuItem>
          <MenuItem>
            <Label>string</Label>
            <Description>string</Description>
            <NavigationPageID>string</NavigationPageID>
            <Childs xsi:nil="true" />
          </MenuItem>
        </MenuItems>
        <Exception>
          <Message>string</Message>
          <Description>string</Description>
          <ExceptionType>SystemException or LogicalException or
            SystemWarning or LogicalWarning</ExceptionType>
        </Exception>
      </Menu>
    </GetMenuResponse>
  </soap:Body>
</soap:Envelope>

```

### 2.3.5 GetPage

### 2.3.6 Description

This method returns a page *exactly as it must be displayed*. Its input parameters are *PageID*, the *id* of the page to display, and data structure *ActionParameters*

```

<ActionParameters>
  <PressedButtonID>string</PressedButtonID>
  <DataForm>Page</DataForm>
  <RecordID>string</RecordID>
  <CallingPageID>string</CallingPageID>
  <AdditionalParameters>Parameter[ ]</AdditionalParameters>
</ActionParameters>

```

where *CallingPageID* contains the *id* of the previous page.

*RecordID* is needed e.g. for an editing page (a page used to edit a record) and provides the *id* of the record to modify. It is a parameter that gets filled when one navigates, e.g. selecting the editing icon in a *grid*.

*DataForm* and *PressedButtonID* have meaning in the context of a *postback*, i.e., when a *data form* is sent from a browser to the server. *DataForm* contains the page as it was, but with updated values of all the controls of all forms reflecting what the user typed in, i.e. the form itself ready to be modified, saved, forwarded and so on. *PressedButtonID* refers to the *id* of the control that has originated the postback: the button, the selected element of the viewed tree, etc.

Method *GetPage* returns an object *Page* with three strings: *ID*, *Title* and *Description* of the page, as well as an array of *PageItems* that represents the different section of a page body: *Form*, with all the controls to be filled by a user (i.e. *FirstName*, *LastName*,...), *GridSection*, *LinkSection*, *TextSection* and *ButtonsSection*. Thanks to the combination of these sections, it is possible to create pages with contents, screens for editing, lists, search pages, and the like. For details about each page type, see Section “4 Data Structures” on page 25.

Object *Page* contains an array of parameters named *PersistingParams*, which may host all those parameters that must be maintained between a postback and another, between a method call and the next, without requiring hidden fields.

### 2.3.6.1 SOAP

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

#### REQUEST:

```
POST /Wsaim/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.icimsi.ch/GetPage"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetPage xmlns="http://www.icimsi.ch">
      <PageID>string</PageID>
      <ActionParameters>
        <PressedButtonID>string</PressedButtonID>
        <DataForm>
          <ID>string</ID>
          <Title>string</Title>
          <Description>string</Description>
          <items>
            <GridSection xsi:nil="true" />
            <Form xsi:nil="true" />
            <TextSection xsi:nil="true" />
            <ButtonsSection xsi:nil="true" />
            <LinksSection xsi:nil="true" />
          </items>
          <PersistingParams>
            <Parameter xsi:nil="true" />
            <Parameter xsi:nil="true" />
          </PersistingParams>
          <Exception>
            <Message>string</Message>
            <Description>string</Description>
          </Exception>
        </DataForm>
      </ActionParameters>
    </GetPage>
  </soap:Body>
</soap:Envelope>
```

```

        <ExceptionType>SystemException or LogicalException or
            SystemWarning or LogicalWarning</ExceptionType>
    </Exception>
</DataForm>
<RecordID>string</RecordID>
<CallingPageID>string</CallingPageID>
<AdditionalParameters>
    <Parameter>
        <Name>string</Name>
        <Value>string</Value>
    </Parameter>
    <Parameter>
        <Name>string</Name>
        <Value>string</Value>
    </Parameter>
</AdditionalParameters>
</ActionParameters>
<AuthenticationData>
    <Logged>boolean</Logged>
    <UserID>string</UserID>
    <UserName>string</UserName>
    <Password>string</Password>
    <GroupID>string</GroupID>
    <AdditionalInformation>
        <Parameter>
            <Name>string</Name>
            <Value>string</Value>
        </Parameter>
        <Parameter>
            <Name>string</Name>
            <Value>string</Value>
        </Parameter>
    </AdditionalInformation>
</AuthenticationData>
</GetPage>
</soap:Body>
</soap:Envelope>

```

**RESPONSE:**

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetPageResponse xmlns="http://www.icimsi.ch">
            <Page>
                <ID>string</ID>
                <Title>string</Title>
                <Description>string</Description>
                <items>
                    <GridSection>
                        <NavigationPageID>string</NavigationPageID>
                        <NavigationParameters xsi:nil="true" />
                        <DataKeyField>string</DataKeyField>
                        <Description>string</Description>
                        <Dataset xsi:nil="true" />
                        <XmlSchemaDefinition xsi:nil="true" />
                    </GridSection>
                </items>
            </Page>
        </GetPageResponse>
    </soap:Body>
</soap:Envelope>

```

```

<Form>
  <ID>string</ID>
  <LastModification />
  <Controls xsi:nil="true" />
</Form>
<TextSection>
  <Text>string</Text>
</TextSection>
<ButtonsSection>
  <Buttons xsi:nil="true" />
</ButtonsSection>
<LinksSection>
  <Label>string</Label>
  <Orientation>Horizontal or Vertical</Orientation>
  <Links xsi:nil="true" />
</LinksSection>
</items>
<PersistingParams>
  <Parameter>
    <Name>string</Name>
    <Value>string</Value>
  </Parameter>
  <Parameter>
    <Name>string</Name>
    <Value>string</Value>
  </Parameter>
</PersistingParams>
<Exception>
  <Message>string</Message>
  <Description>string</Description>
  <ExceptionType>SystemException or LogicalException or
    SystemWarning or LogicalWarning</ExceptionType>
</Exception>
</Page>
</GetPageResponse>
</soap:Body>
</soap:Envelope>

```

## 2.3.7 GetFile

### 2.3.7.1 Description

This method is practically identical to *GetPage*, but instead of returning a *page* it returns a *binary stream*. It has the purpose of passing back, for example, *Word* documents, *Excel* tables, images, *PDF* documents.

The reason for method *GetFile* is that a direct link to a file does not allow integrated control of the access permissions, whereas, thanks to it, it is possible to verify, with *AuthenticationData*, whether the user who is trying to read the file has the proper rights.

It returns an object *File* which contains the following information:

- *Binary data*: the array with the bytes of the file
- *ContentType*: the type of the content, e.g. “*application/pdf*”, “*application/x-zip-compressed*”, “*application/vnd.ms-excel*”, etc.
- *FileName*: the name to give to the file including the extension; the system will propose this name when the use tries to save.

### 2.3.7.2 SOAP

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

#### REQUEST:

```
POST /Wsaim/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.icimsi.ch/GetFile"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetFile xmlns="http://www.icimsi.ch">
      <FileID>string</FileID>
      <ActionParameters>
        <PressedButtonID>string</PressedButtonID>
        <DataForm>
          <ID>string</ID>
          <Title>string</Title>
          <Description>string</Description>
          <items>
            <GridSection xsi:nil="true" />
            <Form xsi:nil="true" />
            <TextSection xsi:nil="true" />
            <ButtonsSection xsi:nil="true" />
            <LinksSection xsi:nil="true" />
          </items>
          <PersistingParams>
            <Parameter xsi:nil="true" />
            <Parameter xsi:nil="true" />
          </PersistingParams>
          <Exception>
            <Message>string</Message>
            <Description>string</Description>
            <ExceptionType>SystemException or LogicalException or
              SystemWarning or LogicalWarning</ExceptionType>
          </Exception>
        </DataForm>
      <RecordID>string</RecordID>
      <CallingPageID>string</CallingPageID>
      <AdditionalParameters>
        <Parameter>
          <Name>string</Name>
          <Value>string</Value>
        </Parameter>
        <Parameter>
          <Name>string</Name>
          <Value>string</Value>
        </Parameter>
      </AdditionalParameters>
    </ActionParameters>
  </soap:Body>
  <AuthenticationData>
    <Logged>boolean</Logged>
    <UserID>string</UserID>
    <UserName>string</UserName>
    <Password>string</Password>
    <GroupID>string</GroupID>
    <AdditionalInformation>
```

```

    <Parameter>
      <Name>string</Name>
      <Value>string</Value>
    </Parameter>
    <Parameter>
      <Name>string</Name>
      <Value>string</Value>
    </Parameter>
  </AdditionalInformation>
</AuthenticationData>
</GetFile>
</soap:Body>
</soap:Envelope>

```

**RESPONSE:**

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: **length**

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetFileResponse xmlns="http://www.icimsi.ch">
      <File>
        <BinaryData>base64Binary</BinaryData>
        <ContentType>string</ContentType>
        <FileName>string</FileName>
        <Exception>
          <Message>string</Message>
          <Description>string</Description>
          <ExceptionType>SystemException or LogicalException or
            SystemWarning or LogicalWarning</ExceptionType>
        </Exception>
      </File>
    </GetFileResponse>
  </soap:Body>
</soap:Envelope>

```

## 2.4 Exception Handling

Exceptions are handled by providing, within every object returned by methods of the Web service, a section *<Exception>*.

```

<Exception>
  <Message>string</Message>
  <Description>string</Description>
  <ExceptionType>SystemException or LogicalException or
    SystemWarning or LogicalWarning</ExceptionType>
</Exception>

```

An *exception* must contain the following information:

- *Message*, the message to display
- *Description*, a description of the error
- *ExceptionType*, the type of error generated: *Logical (Exception or Warning)*; *System (Exception or Warning)*

*Logical exceptions* are those relative to the application, such as “record not found”, “access denied”, etc. *System exceptions* are errors caused by the net infrastructure: timeout, server not found, and the like. For a system exception, the user sees a standard message such as “error so-and-so, the Administrator has been notified”, while for logical exceptions the message is defined by the data structure, such as “The account can not be negative”, “The data you entered has to be ...”, etc.

Messages of type *Warning* are not fatal and only generate a message at the end of an operation, which has been successfully executed all the same (unlike an exception), such as: “This bank transfer is above the limit of 100’000 Euro and must therefore be checked by an authorised clerk. A delay in the execution of the operation may occur.”

## 2.5 Authentication handling

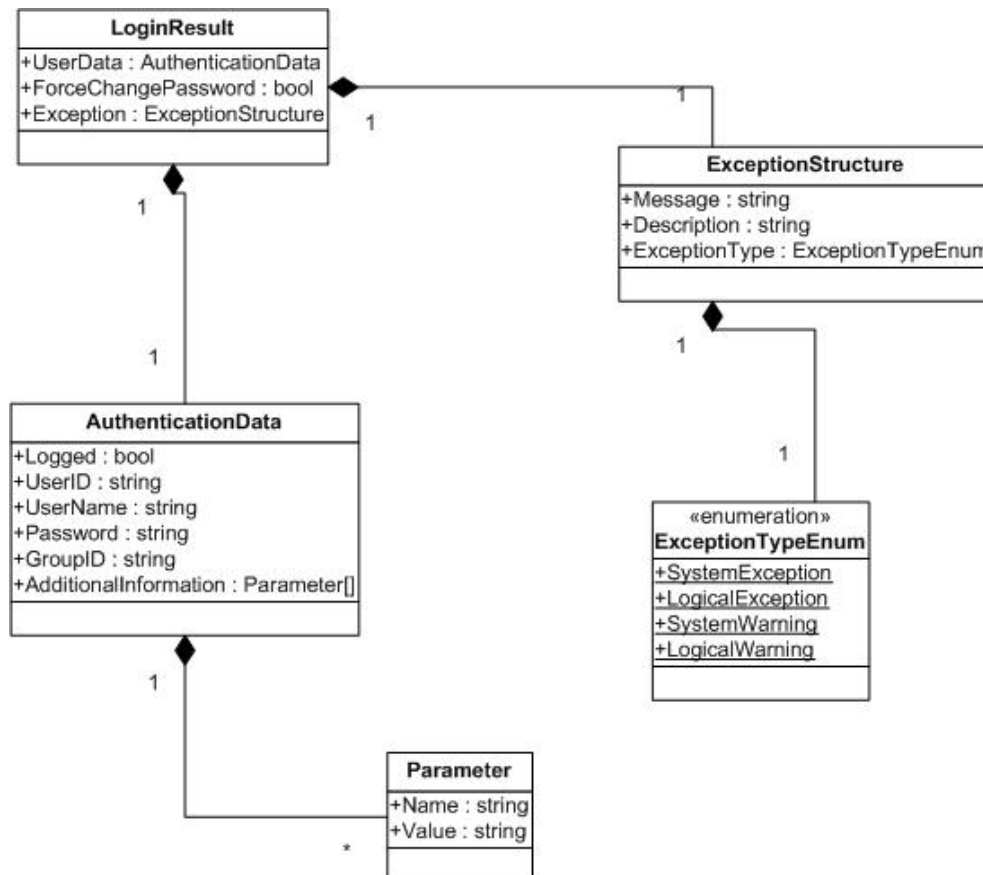
Every method (*GetPage*, *GetFile* and *GetMenu*) always receives an object *AuthenticationData* (filled by method *Login*) with the following information:

```
<AuthenticationData>
  <Logged>boolean</Logged>
  <UserID>string</UserID>
  <UserName>string</UserName>
  <Password>string</Password>
  <GroupID>string</GroupID>
  <AdditionalInformation>
    <Parameter>
      <Name>string</Name>
      <Value>string</Value>
    </Parameter>
    <Parameter>
      <Name>string</Name>
      <Value>string</Value>
    </Parameter>
  </AdditionalInformation>
</AuthenticationData>
```

Therefore, thanks to this structure, it is possible to check whether a user may have access to some data or not, and which operations he/she is allowed to perform.

## 2.6 Diagram of the Authentication Classes

This is the diagram of all the classes involved in the authentication process. The first step is to log into the system by calling the *Login* method that will return object *LoginResult*, in which there is an object *AuthenticationData* that represents the authentication information of the logged user. This data structure has to be passed to every further method call (i.e., *GetPage*, *GetMenu*,...) enabling the web service to verify access rights of the user and return the appropriate records.





## 3 Web Service Interface for Legacy Systems

### 3.1 List of the Methods

#### 3.1.1 Read

Method *Read* is used to read a given record, which will be returned without any decoding, since it is further used for *insert/update/delete* operations.

```
public DataStructure ReadItem(string ID, [other parameters...], string
    UserName, string Password)
```

#### 3.1.2 ReadDecoded

It is used to read a given record that will be displayed to the user, after proper decoding (*SQL join*, etc).

```
public DataStructure ReadDecodedItem(string ID, [other parameters...],
    string UserName, string Password)
```

#### 3.1.3 List

It returns list of *records* ready to be displayed, e.g. within a grid.

```
public DataStructure ListItem([parameters...],
    string UserName, string Password)
```

#### 3.1.4 ListForUpdate

It is similar to method *List*, i.e. it returns a list of *record*, however no decoding takes place, because here data are read for updating multiple records or for deleting in cascade.

#### 3.1.5 Save

This method has the purpose of saving the data set it receives. It would be desirable, of course, to first perform all the necessary checks such as concurrency violation, data integrity, business logic rules, and so on.

If the save operation succeeds, it does not return anything, otherwise it raises an exception:

```
public ExceptionStructure SaveItem(DataStructure DS, string UserName,
    string Password)
```

## 3.2 Exception Handling

The exception handling in the context of the Web Services for legacy systems is the same described in section “2.4 Exception Handling” on page 20.

## 3.3 Authentication handling

Authentication is implemented by passing *UserName* and *Password* to all methods in order to verify in each call whether it is possible or not to return the requested records, or to perform the requested operation.

## 4 Data Structures

In this section, all data structures involved with the interface to XTra.NET are described, by listing the source code with all the comments, together with XML examples of what has to be returned by the Web service.

### 4.1 ExceptionTypeEnum

```
//List of exception types
public enum ExceptionTypeEnum
{
    //System exception that contain information about an error
    //that has not to be shown to the user (i.e.: database
    //error, division by zero, disk error,...). The requested operation has
    //not been completed
    SystemException,
    //Logical (related to the program logic) exception that has
    //to be shown to the user (i.e.: "the balance of this account cannot be
    //less than zero", "invalid password",...). The requested operation has
    //not been completed
    LogicalException,
    //This exception has not to be notified to the user. (i.e.:
    //"the disk usage has reached the 90%, in a short time the
    //disk will be full"). The requested operation has been
    //completed.
    SystemWarning,
    //This exception has to be notified to the user. (i.e.:
    //"you are reaching the account limit", "the operation has
    //completed but the notification has not been created").
    //The requested operation has been completed.
    LogicalWarning
}
```

### 4.2 ExceptionStructure

```
public class Exception
{
    //Error message
    public string Message;
    //Error description (line of code where the exception was
```

```

//thrown, file name, call stack,...)
public string Description;
//Type of exception
public ExceptionTypeEnum ExceptionType;
}

```

### 4.2.1 Example

```

<Exception>
  <Message>
    Record Expired
  </Message>
  <Description>
    The record is expired. You will not be able to modify it
  </Description>
  <ExceptionType>LogicalWarning</ExceptionType>
</Exception>

```

## 4.3 DataStructure

```

public class DataStructure
{
  //This is the schema definition of the XmlDataStructure
  public XmlDocument XmlSchemaDefinition;
  //This property contains the data
  public XmlDocument XmlDataStructure;
  //This is an optional property that contains the occurred
  //exception
  public ExceptionStructure Exception;
}

```

### 4.3.1 Example

```

<?xml version="1.0" encoding="utf-8"?>
<DataStructure xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.icimsi.ch">
  <XmlSchemaDefinition>
    <xs:schema id="FormStructure" xmlns=""
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
      <xs:element name="FormStructure" msdata:IsDataSet="true"
        msdata:Locale="it-CH">
        <xs:complexType>
          <xs:choice maxOccurs="unbounded">
            <xs:element name="TableRows">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string" />
                  <xs:element name="PageID" type="xs:string" />
                  <xs:element name="col2" type="xs:string"
                    minOccurs="0" />
                  <xs:element name="col3" type="xs:string"
                    minOccurs="0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:element>
      </xs:schema>
    </XmlSchemaDefinition>
  </DataStructure>

```

---

```

        </xs:element>
    </xs:choice>
</xs:complexType>
<xs:unique name="Constraint1">
    <xs:selector xpath="//TableRows" />
    <xs:field xpath="ID" />
</xs:unique>
<xs:unique name="Constraint2">
    <xs:selector xpath="//TableRows" />
    <xs:field xpath="PageID" />
</xs:unique>
</xs:element>
</xs:schema>
</XmlSchemaDefinition>
<XmlDataStructure>
    <FormStructure xmlns="">
        <TableRows>
            <ID>341421</ID>
            <PageID>56456</PageID>
            <col2>Comumn 1</col2>
            <col3>Comumn 2</col3>
        </TableRows>
    </FormStructure>
</XmlDataStructure>
<Exception>
    <Message>
        Record Expired
    </Message>
    <Description>
        The record is expired. You will not be able to modify it
    </Description>
    <ExceptionType>LogicalWarning</ExceptionType>
</Exception>
</DataStructure>

```

## 4.4 Page

```

public class Page
{
    //The ID of the page
    public string ID;

    //Title to be displayed
    public string Title;

    //Optional description to display near the title
    public string Description;

    //The content of the page
    public PageItem[] items;

    //Information transferred between calls
    public Parameter[] PersistingParams;

    //Exception (if necessary)
    public ExceptionStructure Exception;
}

```

### 4.4.1 Example 1 - Documents Browser

```

<?xml version="1.0" encoding="utf-8"?>
<Page xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.icimsi.ch">
  <ID>mdl_documents_list</ID>
  <Title>Documents</Title>
  <Description>This is a documents browser page.</Description>
  <items>
    <Form>
      <ID>frmDir</ID>
      <Controls>
        <TreeView>
          <ID>treDir</ID>
          <Description>Directory Tree</Description>
          <Label>Directory</Label>
          <ReadOnly>>false</ReadOnly>
          <SelectedNodeID>0</SelectedNodeID>
          <Nodes>
            <TreeNode>
              <ID>0</ID>
              <Text>Root</Text>
              <NavigationPageID>MDL_Documents_List</NavigationPageID>
              <Childs>
                <TreeNode>
                  <ID>8</ID>
                  <Text>1</Text>
                  <NavigationPageID>
                    MDL_Documents_List
                  </NavigationPageID>
                  <Childs>
                    <TreeNode>
                      <ID>15</ID>
                      <Text>1.1</Text>
                      <NavigationPageID>
                        MDL_Documents_List
                      </NavigationPageID>
                      <Childs>
                        <TreeNode>
                          <ID>16</ID>
                          <Text>1.1.1</Text>
                          <NavigationPageID>
                            MDL_Documents_List
                          </NavigationPageID>
                          <Childs />
                        </TreeNode>
                      </Childs>
                    </TreeNode>
                  </Childs>
                </TreeNode>
              </Childs>
            </TreeNode>
            <TreeNode>
              <ID>14</ID>
              <Text>10+</Text>
              <NavigationPageID>
                MDL_Documents_List
              </NavigationPageID>
              <Childs />
            </TreeNode>
            <TreeNode>
              <ID>12</ID>
              <Text>2</Text>
              <NavigationPageID>
                MDL_Documents_List
              </NavigationPageID>
              <Childs />
            </TreeNode>
            <TreeNode>
              <ID>13</ID>

```

```

        <Text>3</Text>
        <NavigationPageID>
            MDL_Documents_List
        </NavigationPageID>
        <Childs />
    </TreeNode>
</Childs>
</TreeNode>
</Nodes>
</TreeView>
</Controls>
</Form>
<LinksSection>
    <Label>Actions</Label>
    <Orientation>Horizontal</Orientation>
    <Links>
        <Link>
            <ID>lnkNewFile</ID>
            <Label>New File</Label>
            <Description>Add New File</Description>
            <NavigationType>Page</NavigationType>
            <NavigationID>MDL_Documents_Edit</NavigationID>
            <NavigationRecordID>0</NavigationRecordID>
            <NavigationParameters>
                <Parameter>
                    <Name>Directories_ID</Name>
                    <Value>0</Value>
                </Parameter>
            </NavigationParameters>
        </Link>
        <Link>
            <ID>lnkNewDir</ID>
            <Label>New Directory</Label>
            <Description>New SubDirectory</Description>
            <NavigationType>Page</NavigationType>
            <NavigationID>MDL_Directories_Edit</NavigationID>
            <NavigationRecordID>0</NavigationRecordID>
            <NavigationParameters>
                <Parameter>
                    <Name>Parent_ID</Name>
                    <Value>0</Value>
                </Parameter>
            </NavigationParameters>
        </Link>
    </Links>
</LinksSection>
<GridSection>
    <NavigationPageID>MDL_Documents_Actions</NavigationPageID>
    <DataKeyField>ID</DataKeyField>
    <Description>Documents List</Description>
    <Dataset>
        <NewDataSet xmlns="">
            <Table>
                <ID>142</ID>
                <Title>fdasadsf</Title>
                <Description>adsfdsfasdf</Description>
                <Size>1175552</Size>
                <Directories_ID>0</Directories_ID>
                <Status />
                <Visibility>Standard</Visibility>
            </Table>
            <Table>
                <ID>140</ID>
                <Title>My DVD Serial</Title>
                <Description>fdaadsf</Description>
                <Size>3608</Size>
                <Directories_ID>0</Directories_ID>
            </Table>
        </NewDataSet>
    </Dataset>
</GridSection>

```

```

    <Status />
    <Visibility>Restricted</Visibility>
  </Table>
  <Table>
    <ID>141</ID>
    <Title>test</Title>
    <Description>dsfaadsf</Description>
    <Size>200704</Size>
    <Directories_ID>0</Directories_ID>
    <Status />
    <Visibility>Standard</Visibility>
  </Table>
</NewDataSet>
</Dataset>
<XmlSchemaDefinition>
  <xs:schema id="NewDataSet" xmlns=""
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="NewDataSet" msdata:IsDataSet="true"
      msdata:EnforceConstraints="False">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Table">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ID" type="xs:int"
                  minOccurs="0" />
                <xs:element name="Title" type="xs:string"
                  minOccurs="0" />
                <xs:element name="Description" type="xs:string"
                  minOccurs="0" />
                <xs:element name="Size" type="xs:int"
                  minOccurs="0" />
                <xs:element name="Directories_ID" type="xs:int"
                  minOccurs="0" />
                <xs:element name="Status" type="xs:string"
                  minOccurs="0" />
                <xs:element name="Visibility" type="xs:string"
                  minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
</XmlSchemaDefinition>
</GridSection>
</items>
<PersistingParams>
  <Parameter>
    <Name>Directories_ID</Name>
    <Value>0</Value>
  </Parameter>
</PersistingParams>
</Page>

```

#### 4.4.2 Example 2 - Links Edit

```

<?xml version="1.0" encoding="utf-8"?>
<Page xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

    xmlns="http://www.icimsi.ch">
<ID>mdl_linksmanagement_edit</ID>
<Title>Edit Links</Title>
<Description>This is a links edit page.</Description>
<items>
  <Form>
    <ID>frmLink</ID>
    <LastModification xsi:type="xsd:base64Binary">
      AAAAAAAAAUA=</LastModification>
    <Controls>
      <TextBoxString>
        <ID>Name</ID>
        <Label>Name</Label>
        <ReadOnly>>false</ReadOnly>
        <RequiredField>>true</RequiredField>
        <MaxLength>50</MaxLength>
        <Value>Europa</Value>
      </TextBoxString>
      <TextBoxString>
        <ID>Description</ID>
        <Label>Description</Label>
        <ReadOnly>>false</ReadOnly>
        <RequiredField>>false</RequiredField>
        <MaxLength>1000</MaxLength>
        <Value>Portale dell'Unione Europea</Value>
      </TextBoxString>
      <TextBoxString>
        <ID>Link</ID>
        <Label>Link</Label>
        <ReadOnly>>false</ReadOnly>
        <RequiredField>>true</RequiredField>
        <MaxLength>200</MaxLength>
        <ValidationRegExp>
          http://([\w-]+\.)+[\w-]+(\/[\w- .\/?%&#;=]*)?
        </ValidationRegExp>
        <Value>http://europa.eu.int/index-it.htm</Value>
      </TextBoxString>
      <DropDownList>
        <ID>public_private</ID>
        <Label>Visibility</Label>
        <ReadOnly>>false</ReadOnly>
        <RequiredField>>true</RequiredField>
        <Options>
          <Option>
            <Value>0</Value>
            <Text>Standard</Text>
          </Option>
          <Option>
            <Value>1</Value>
            <Text>Restricted</Text>
          </Option>
        </Options>
        <Value>0</Value>
        <AutoPostBack>>false</AutoPostBack>
      </DropDownList>
    </Controls>
  </Form>
  <ButtonsSection>
    <Buttons>
      <Button>
        <ID>Save</ID>
        <Label>Save</Label>
        <Description>Save the data</Description>
        <CausesFormValidation>>true</CausesFormValidation>
      </Button>
      <Button>
        <ID>Delete</ID>

```

```

    <Label>Delete</Label>
    <Description>Deletes the record</Description>
    <CausesFormValidation>>false</CausesFormValidation>
    <ConfirmationMessage>
        Are you sure to delete this record?
    </ConfirmationMessage>
  </Button>
</Buttons>
</ButtonsSection>
</items>
</Page>

```

### 4.4.3 Example 3 – Documents Edit

```

<?xml version="1.0" encoding="utf-8"?>
<Page xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.icimsi.ch">
  <ID>mdl_documents_edit</ID>
  <Title>Edit Documents</Title>
  <Description>This is a document edit page.</Description>
  <items>
    <Form>
      <ID>frmDocuments</ID>
      <LastModification xsi:type="xsd:base64Binary">
        AAAAAAABAQk=
      </LastModification>
      <Controls>
        <HiddenField>
          <ID>Directories_ID</ID>
          <ReadOnly>>false</ReadOnly>
          <Value xsi:type="xsd:string">9</Value>
        </HiddenField>
        <TextBoxString>
          <ID>Title</ID>
          <Label>Title</Label>
          <ReadOnly>>false</ReadOnly>
          <RequiredField>>true</RequiredField>
          <MaxLength>50</MaxLength>
          <Value>Test</Value>
        </TextBoxString>
        <TextBoxString>
          <ID>Description</ID>
          <Label>Description</Label>
          <ReadOnly>>false</ReadOnly>
          <RequiredField>>false</RequiredField>
          <MaxLength>1000</MaxLength>
          <Value>This is a test document</Value>
        </TextBoxString>
        <TextBoxString>
          <ID>Keywords</ID>
          <Label>Keywords</Label>
          <ReadOnly>>false</ReadOnly>
          <RequiredField>>true</RequiredField>
          <MaxLength>200</MaxLength>
          <Value>XTra.NET AIM Test Web Service</Value>
        </TextBoxString>
        <InputFile>
          <ID>File</ID>
          <Label>File</Label>
          <ReadOnly>>true</ReadOnly>
          <RequiredField>>true</RequiredField>
          <Size>6385</Size>
        </InputFile>
      </Controls>
    </Form>
  </items>
</Page>

```

```

<DropDownList>
  <ID>public_private</ID>
  <Label>Visibility</Label>
  <ReadOnly>>false</ReadOnly>
  <RequiredField>>true</RequiredField>
  <Options>
    <Option>
      <Value>0</Value>
      <Text>Standard</Text>
    </Option>
    <Option>
      <Value>1</Value>
      <Text>Restricted</Text>
    </Option>
  </Options>
  <Value>0</Value>
  <AutoPostBack>>false</AutoPostBack>
</DropDownList>
</Controls>
</Form>
<ButtonsSection>
  <Buttons>
    <Button>
      <ID>btnCheckout</ID>
      <Label>Checkout</Label>
      <Description>
        Checkout current document for modification
      </Description>
      <CausesFormValidation>>false</CausesFormValidation>
    </Button>
  </Buttons>
</ButtonsSection>
<ButtonsSection>
  <Buttons>
    <Button>
      <ID>Save</ID>
      <Label>Save</Label>
      <Description>Save the data</Description>
      <CausesFormValidation>>true</CausesFormValidation>
    </Button>
    <Button>
      <ID>Delete</ID>
      <Label>Delete</Label>
      <Description>Deletes the record</Description>
      <CausesFormValidation>>false</CausesFormValidation>
      <ConfirmationMessage>
        Are you sure to delete this record?
      </ConfirmationMessage>
    </Button>
  </Buttons>
</ButtonsSection>
</items>
</Page>

```

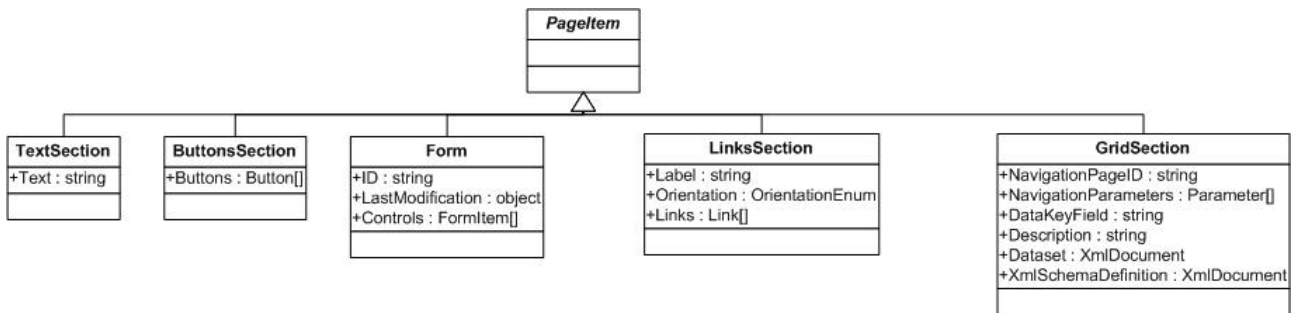
## 4.5 PageItem

```

//Abstract class defining the page item
public abstract class PageItem
{
}

```

## 4.5.1 Class Diagram



## 4.6 TextSection

```

//A text section is displayed to the user in the
//page.
public class TextSection: PageItem
{
    public string Text;
}
  
```

### 4.6.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.7 GridSection

```

//This section is displayed to the user as a grid
public class GridSection: PageItem
{
    //Page ID to be sent to the webservice when a row is
    //selected
    public string NavigationPageID;

    //Any additional parameter to be sent to the webservice
    //when a row is selected
    public Parameter[] NavigationParameters;

    //Field of the Data Set that represents the key to be
    //passed back to the webservice at the moment of the
    //selection of a record
    public string DataKeyField;

    //Description of what the grid represents
    public string Description;

    //The data to be displayed into the grid
    public XmlDocument Dataset;

    //Schema of the DataSet
    public XmlDocument XmlSchemaDefinition;
}
  
```

### 4.7.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.8 Buttonsection

```
//This section contains the buttons to be displayed to the user
public class ButtonSection: PageItem
{
    public Button[] Buttons;
}
```

### 4.8.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.9 Button

```
public class Button
{
    //Identifier of the button. This identifier is used by the
    //webservice to determine what to do when the button is
    //pressed
    public string ID;

    //Caption of the button
    public string Label;

    //The description showed as a tooltip
    public string Description;

    //If true, when the user presses it, the XTra.NET framework
    //will do all the validation on the form (mandatory fields,
    //regular expression,...) on the client side. For example
    //if it is necessary to check if all the fields are valid
    //when the user presses the "Save" button but it is not
    //necessary to do it if he presses the "Cancel" button you
    //need to set this property to true for the "Save" button
    //and to false for the "Cancel" button
    public bool CausesFormValidation;

    //This message is displayed to the user at the moment he
    //presses the button with a confirmation message box (i.e.:
    //"Are you sure you want to delete this item?")
    public string ConfirmationMessage;
}
```

### 4.9.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.10 LinkSection

```
public class LinksSection: PageItem
{
    //Label for this link section (optional)
    public string Label;

    //Orientation: Orizontal or Vertical
    public OrientationEnum Orientation;

    //Array of links
    public Link[] Links;
}
```

### 4.10.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.11 Link

```
public class Link
{
    public string ID;

    //Label to display
    public string Label;

    //Description showed as a tooltip
    public string Description;

    //Navigation type: Page or File. Specify if the linked
    //object is another page or a file. This property is used
    //to determine the method to be called (GetPage or
    //GetFile).
    public NavigationTypeEnum NavigationType;

    //ID of the Page or the File to navigate to
    public string NavigationID;

    //Optional ID of the record to navigate to
    public string NavigationRecordID;

    //Any additional parameter to be passed to the web service
    public Parameter[] NavigationParameters;
}
```

### 4.11.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.12 NavigationTypeEnum

```
//Navigation type: Page or File. Specify if the linked
//object is another page or a file. This property is used
//to determine the method to be called (GetPage or
//GetFile).
public enum NavigationTypeEnum
{
    Page, File
}
```

## 4.13 OrientationEnum

```
public enum OrientationEnum
{
    Horizontal, Vertical
}
```

## 4.14 Form

```
public class Form : PageItem
{
    public string ID;

    //This is an information that allows the web service to
    //manage the concurrency violation (2 users modifying the
    //same record). The web service has to read the record from
    //the database and compare the LastModification field: if
    //it is changed it means that another user has modified the
    //record. If no concurrency violation check is needed this
    //field can be set to null.
    public object LastModification;

    //The list of controls of the form
    public FormItem[] Controls;
}
```

### 4.14.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.15 FormItem

```
//Abstract class that defines the structure of all FormItem
//derived classes
public abstract class FormItem
{
    //Unique identifier that will be used by the web service to
    //identify the items (button → identify the event
    //triggered, textbox → determine what value has to be
```

---

```

//copied in which database field
public string ID;

//Description of the item (used as an help)
public string Description;

//Text to be displayed to the user to identify what the
//item is (i.e.: "last name", "first name", ...)
public string Label;

//Determines if the field is enabled or disabled (readonly)
public bool ReadOnly;
}

```

## 4.16 HiddenField

```

//Used to store information in the page without the user to be
//able to see them. This is necessary because the internet model
//is state less and for that reason if you need to simulate the
//statefull model the only way is to store hidden information in
//the page.

//I.e.: if you have to check the concurrency violation you need
//to store the last modification information (a timestamp) of the
//original record but you cannot do it on the server because you
//will loose this information due to the fact the internet model
//is state less. So if you what to store this information the
//only way is to store it in the page and when the form is posted
//back again you will have it in the form object.
public class HiddenField : FormItem
{
    //Value to be stored
    public object Value;
}

```

### 4.16.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.17 TextBox

```

//This is the abstract class that defines the basic behavior of a
//generic text box
public abstract class TextBox : FormItem
{
    //Indicates if the field is required
    public bool RequiredField;

    //Indicated the maximum length of the data the user can
    //type in the text box
    public int MaxLength;

    //This is the regular expression for the validation of the
    //value
    public string ValidationRegExp;
}

```

```
//The value. This field is used as input and output value
//If the webservice specifies it, it will be
//shown as initial value of the textbox. This property will
//be set with the value entered by the user at the moment
//the webservice is called back
public string Value;
}
```

#### 4.17.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

### 4.18 TextBoxString

```
//This is a textbox that allows the user to enter text (string
//format)
public class TextBoxString : TextBox
{
}
```

#### 4.18.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

### 4.19 TextBoxNumber

```
//This is a textbox that allows the user to enter a number
//(numeric format)
public class TextBoxNumber : TextBox
{
}
```

#### 4.19.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

### 4.20 InputDate

```
//This control allows the user to enter a date into the form
public class InputDate : FormItem
{
    //Indicates if the field is required
    public bool RequiredField;

    //The value. This field is used as input and output value
}
```

---

```

//If the webservice specifies it, it will be
//shown as initial value of the textbox. This property will
//be set with the value entered by the user at the moment
//the webservice is called back
public DateTime Value;
}

```

#### 4.20.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

#### 4.21 InputFile

```

//This control allows the user to upload a file
public class InputFile : FormItem
{
    //Indicates if the field is required
    public bool RequiredField;

    //The Regular Expression to control the file type to
    //upload (the check will be based on file extension)
    public string FileTypeRegExp;

    //The size of the file uploaded.
    public int Size;

    //The type of the file ("application/pdf",
    //"application/x-zip-compressed",
    //"application/vnd.ms-excel", ...)
    public string ContentType;

    //The name of the file uploaded (with extension)
    public string FileName;

    //The byte array representing the file uploaded
    public byte[] BinaryData;
}

```

NB. When the Web service returns a *form* with data with a field of type *InputFile*, to indicate to XTra.Net that the file has already been uploaded, it is sufficient to specify a size different from 0. (notice that XTra.Net will modify binary data only if the user decided to specify a file which is not the one uploaded previously). This has been designed to reduce the net load, since it is not necessary to also return *BinaryData*, which may have a large size, several Mb.

#### 4.21.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.22 Option

```
//Describes the element of a multiple choice control type
public class Option
{
    //The string to be shown to the user to allow him/her to
    //choose (i.e.: country like Spain, France, Italy,...)
    public string Text;

    //The value that is associated to the option (i.e.: the
    //unique identifier of the record in the database, Spain =
    //123, France = 654,...)
    public string Value;
}
```

## 4.23 MultipleChoice

```
//Abstract class defining a multiple choice control (such as a
//listbox, a dropdownlist, a radiobutton list,...)
public abstract class MultipleChoice : FormItem
{
    //Indicates if the field is required
    public bool RequiredField;

    public Option[] Options;

    //The value to be marked as selected or the value the user
    //selected.
    //This field is used as input and output value. If the
    //webservice specifies it, it will be
    //shown as selected value in the list. This property will
    //be set with the value entered by the user at the moment
    //the webservice is called back
    public string Value;
}
```

## 4.24 DropDownList

```
public class DropDownList : MultipleChoice
{
    //If set to true, at the moment the user selects a value
    //the webservice is called automatically without the need
    //the user presses a button
    public bool AutoPostBack;
}
```

### 4.24.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.25 RadioButtons

```
public class RadioButtons : MultipleChoice
{
}
```

### 4.25.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.26 TreeView

```
public class TreeView : FormItem
{
    //The id of the selected node
    public string SelectedNodeID;

    //Array of TreeNode
    public TreeNode[] Nodes;
}
```

### 4.26.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.27 TreeNode

```
public class TreeNode
{
    //ID of this node
    public string ID;

    //Label to display for this node
    public string Text;

    //The page to navigate to on click (if specified XTra.NET
    //will navigate to the page specified on the user click, if
    //not specified XTra.NET will call the Web Service using
    //the same page id)
    public string NavigationPageID;

    //Child elements of this node.
    public TreeNode[] Childs;
}
```

### 4.27.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.28 CheckBox

```
public class CheckBox : FormItem
{
    //Checked or not
    //This field is used as input and output value
    //If the webservice specifies it, it will be
    //shown as initial value of the checkbox. This property
    //will be set with the value entered by the user at the
    //moment the webservice is called back
    public bool Value;
}
```

### 4.28.1 Example

See the *Page* structure description in Section “4.4 Page” on page 27.

## 4.29 Menu

```
public class Menu
{
    public MenuItem[] MenuItems;
}
```

### 4.29.1 Example

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfMenuItem xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.icimsi.ch">
  <MenuItem>
    <Label>Menu 1</Label>
    <Description>[Container]</Description>
    <Childs>
      <MenuItem>
        <Label>Form 'F31CH3R-FRM01'</Label>
        <Description>Go To Form 'F31CH3R-FRM01'</Description>
        <NavigationPageID>F31CH3R-FRM01</NavigationPageID>
      </MenuItem>
      <MenuItem>
        <Label>Sample List 'F31CH3R-LST01'</Label>
        <Description>List 'F31CH3R-LST01'</Description>
        <NavigationPageID>F31CH3R-LST01</NavigationPageID>
      </MenuItem>
    </Childs>
  </MenuItem>
  <MenuItem>
    <Label>Menu 2</Label>
    <Description>[Container]</Description>
    <Childs>
      <MenuItem>
        <Label>Form 'F31CH3R-FRM03'</Label>
        <Description>Go To Form 'F31CH3R-FRM03'</Description>
        <NavigationPageID>F31CH3R-FRM03</NavigationPageID>
      </MenuItem>
    </Childs>
  </MenuItem>
</ArrayOfMenuItem>
```

```
</MenuItem>
<MenuItem>
  <Label>Sample List 'F31CH3R-LST06'</Label>
  <Description>List 'F31CH3R-LST06'</Description>
  <NavigationPageID>F31CH3R-LST06</NavigationPageID>
</MenuItem>
</Childs>
</MenuItem>
</ArrayOfMenuItem>
```

## 4.30 MenuItem

```
public class MenuItem
{
    //Text to be displayed
    public string Label;

    //Description showed as a tooltip
    public string Description;

    //The page to navigate to
    public string NavigationPageID;

    //Childs of this menu, sub-menus
    public MenuItem[] Childs;
}
```

### 4.30.1 Example

See the *Menu* structure description in Section “4.29 Menu” on page 43.

## 5 Conclusion

This document is the complete description of the specification for the development of a Web Service for XTra.NET Interfacing, the system that will provide Web access to the AIM system.

In addition to this document, there exist:

1. *a complete example of a web service*, developed using Microsoft Visual Studio .NET; this example is an implementation of a web interface for a *document manager*, a *link manager* and a *content manager* (the visualisation part)
2. the complete *Web Service Definition Language*, or WSDL, based on the specification

These two annexes are of purely technical nature and are not part of this document, but have been sent to those partners who develop the software of AIM. It is claimed that, together with this document, they allow a practitioner of the art to fully develop the interfaces to realise the Web services of this Workpackage.