Annexe: Journal

| JOURNAL W1 |

25.05.09

- Session 1 at EIF C 10 04 with François Buntschu and Philippe Joye --> see protocol
- Research / Information about the product MileGate
- Research on SOAP/ WSDL
- Research on ancient diploma project at KEYMILE

26.05.09

- Introduction and presentation of the enterprise by Daniel Gachet
- Introduction into MileGate by Marcel Bellorini
- Installation of implementaion environment / working place
- Demonstration of Prototype
- Selfstudy of MCST (USB Configuration Tool), SOAPui (Services Test Tool) and Prototype

27.05.09

- Study of document "Introduction on the MileGate XML - Management Interface"
- Study of document "Web Sercives Interface for Milegate"
- Discussion on possible Tasks
- Talk with Marcel Bellorini and Daniel Gachet (Topic: possible tasks)

28.05.09

- Definition of tasks
- Talk with Marcel Bellorini and Daniel Gachet (Topic: acctual list of tasks)
- Discussion of possible tasks
- Clarification of difference between solution using web services and HTML
- Decision to organize an emergancy meeting to discuss the project description

29.05.09

- Meeting with Daniel Gachet and Philippe Joye (Topic: necessity to change the project description?)
  - |- Presentation of the desired system (graphic in task book)
  - |- Differentiation Machine-Machine Interface and Human-Machine Interface
  - |- Communication of focuses for the project
  - |- Decision: no necessity to change the description
- Definition of tasks

02.06.09

- Modification of task book / planning
- create graphics
- Website (project info)
- Survey on HTTP Servers / HTML generation

| JOURNAL W2 |

03.06.09 - 09.06.09

- Session 2 at KEYMILE with François Buntschu, Daniel Gachet and Philippe Joye --> see protocol
- Talk with Daniel Gachet (Topic: adaptation of schema)
  - |- Split of schema HTML Service & Web Service
  - |- Add comparative Schema for actual management interface (KOAP and CLI)
- Discussed modifications on the Task book
- MILESTONE (03.06.09): Delivery of Task book & Planning

- Overview of web services (WSDL Description & SOAP Request/ Response message)

HTML Service (David Schneider):

- Survey of acctual management system MCST
- Study of document "User Guide - MileGate & MCST" chapter 1-6
- Definition of structure
- Analysis of limitations, communication with MileGate and functionality of the management system
- Identification of problems (feasability) including some discussions with Marcel Bellorini
- Elaboration of a recommendation

Embedded HTTP servers (Thierry KIKI):

- Analysis of Milegate constraints (CPU usage memory limitation...)
- Proposal of a list of embedded http servers
- Definition of required fields and selective criteria
- Discussion with Marcel about KEYMILE requirements and proposal of a comparative study contents
- Overview and deep study of features of each proposed embedded HTTP Serve : Unfinished
- Comparative study of a few embedded http servers : Unfinished
- Proposal of solutions (suggestions) : Unfinished & Coming soon as report

| JOURNAL W3 |

10.06.09 - 16.06.09

- Session 3 at KEYMILE with François Buntschu, Daniel Gachet and Philippe Joye --> see protocol
- Correction/modification of documents according to the session

Web Services:

- Introduction into Web Services and Service Oriented Architectures
- Description language (WSDL), messages (SOAP) and directory service (UDDI)
- Additional WS concepts (WS-Addressing, WS-Transfer, WS-Management, WS-Notification, ..)
- Observation of actual system

Client Framework:

- Research of Frameworks
- First installation

| JOURNAL W4 |

17.06.09 - 23.06.09

- no Session

Web Services Description (David Schneider):

- Further survey of WS-Addressing/ EndpointReference concept, WS-Transfer, WS-Management, WS-Distributed Management(MUWS 1.0)
- The aim of the studies to approach the discovery of the tree and with that the proper addressing of the MO's (ManagedObject)
- Automatical write of SOAP Headers from WSDL
- Effect of WSDL changement (SOAPui, Client Framework)
- Installation of Axis2

Web Services : (Thierry KIKI):

- Survey WSDL & SOAP
- Difference between eache Version (WSDL 1.0/1.1/2.0 and SOAP 1.1/1.2)
- Survey WS-* : Addressing, Tranfer (Not in deeper)
- SOAP : Type of message and encoding
- Web Services Interoperability

Client Framework : (Thierry KIKI)

- Client Structure, look for IDE
- Installation of Axis2 java and C++ (don't work ?!)
- Installation of Visual Studio 2008 (Not yet : license problem with KEYMILE)
- Defining what to be evaluate on client tools.

| JOURNAL W 5-6 |

24.06.09 - 06.07.09

- Session 4 at KEYMILE with Daniel Gachet and Philippe Joye --> see protocol

- Definition of validation/testing for Web Service
  Necessary changes for prototype
- Preparation of final report
- Text and Images for Flyer

Web Services Description (David Schneider):

- Documentation of Web Service concepts
- Finalize Web Service Description
- Testing of WSDL with SOAPui
- Description of Milegate (Documentation)

Client Framework : (Thierry KIKI)

- Installation of Visual Studio
- Installation of AXIS2
- Completion of Framework study
- Documentation of Embedded Servers

| JOURNAL W7 |

24.06.09 - 06.07.09

- Session 5 at KEYMILE with François Buntschu, Daniel Gachet and Philippe Joye --> see protocol
- Documentation

Web Services (David Schneider):

- Automatical generation of WSDL file with XSLT performed and documented
- Verification of Web Service as defined
- Flow of information documented

Client Framework : (Thierry KIKI)

- Installation of module xFer (WS-Transfer) for Axis
- Definition of tests
- Generation of JAVA classes

Annexe: PV

KEYMILE access to the world	Manage telecommunication using Web Services Protocol 25.05.2009 / 08h30 – 10h05	Seite / Page 1/1 Next session date **03.06.2009**

Students:	**EIF Profs**:	**Copy to** :
Thierry KIKI ✓ David SCHNEIDER ✓	Philippe JOYE ✓ François BUNTSCHU ✓ **KEYMILE :** Daniel GACHET (Mandatory) Marcel BELLORINI	N. MAYENCOURT (Expert) ✓

LIST OF POINTS:

1. Delivery of project description
   - Overall presentation of the project
   - Explanation on possibilities / topics
   - Work to perform

2. Organizational details
   - Transport
   - Lunch
   - Appointment with M. Daniel to know the list of references which will be used.

3. Responsibilities
   - NDA (Source codes will be kept inside KEYMILE walls and so one…)
   - Respect the confidential clause (Confidential documents)

4. Expected results & how the project will be evaluated.

JOB FOR THE NEXT SESSION:

1. Tasks book
2. Planning of project
3. An accurate tasks splitting
4. List of references
5. Divers

KEYMILE access to the world	Manage telecommunication using Web Services Protocol 03.06.2009 / 11h00 – 12h00	Seite / Page 1/1 Next session date **10.06.2009**

Students:	**EIF Profs**:	**Copy to** :
Thierry KIKI ✓	Philippe JOYE ✓	
David SCHNEIDER ✓	François BUNTSCHU ✓	
	KEYMILE :	N. MAYENCOURT (Expert) ✓
	Daniel GACHET (Mandatory) ✓	Marcel BELLORINI ✓

LIST OF POINTS:

1. Task book amelioration & validation
   - Be more accurate about the final task book' objectives definition.
   - among the objectives, find which ones have high /low priority according to KEYMILE requirements
   - Specify on the schema the varied objectives.
   - Nomenclature of pictures, spelling & syntax mistake…
   - Structure of schemas

2. Weekly job presentation
   - Summary to M. Buntschu
   - Weekly information : logbook (from 25-05-09 to 02-06-09)

3. Planning
   - Task splitting
   - Planning validation

JOB FOR THE NEXT SESSION:

1. Result of embedded web server surveying.
2. Analysis of HTML generation service.
3. Divers

KEYMILE access to the world	**Manage telecommunication using Web Services** **Protocol** 10.06.2009 / 13h30 – 15h00	Seite / Page 1/1 Next session date **not fixed**

Students:	**EIF Profs**:	**Copy to** :
Thierry KIKI ✓	Philippe JOYE ✓	
David SCHNEIDER ✓	François BUNTSCHU ✓	
	KEYMILE :	N. MAYENCOURT (Expert) ✓
	Daniel GACHET (Mandatary) ✓	Marcel BELLORINI ✓

LIST OF POINTS:

1. Summary of last session

2. Presentation of the last week
   - Journal

3. Result of embedded web server surveying.
   - Correction of the structure
   - Separation of the personal and KEYMILE requirements

4. Analysis of HTML generation service.
   - Presentation of the service
   - Discussion on what will be saved on MileGate
   - Discussion on divers problems

5. Divers
   - Necessity of next weeks session (will be announced by mail)

JOB FOR THE NEXT WEEK:

1. Survey of state of the art of Web Services

2. Correction and completion of documents according to the discussion

3. Divers

David Schneider, 16.06.09

KEYMILE access to the world	**Manage telecommunication using Web Services** **Protocol** 24.06.2009 / 13h45 – 15h30	Seite / Page 1/1 Next session date 06/07/09

Students:		**EIF Profs**:		**Copy to** :	
Thierry KIKI	✓	Philippe JOYE	✓	N. MAYENCOURT (Expert)	✓
David SCHNEIDER	✓	**KEYMILE :**		Marcel BELLORINI	✓
		Daniel GACHET (Mandatary)	✓	François BUNTSCHU	✓

LIST OF POINTS:

1. Presentation of the last two week
   - Journal

2. Presentation of the Web Service
   - Presentation of the WebService Description (WSDL)
   - Presentation of the interesting Web Service concept
   - Discussion: Concepts to survey further
   - Information about exigence on Frameworks

3. Presentation of interoperability
   - Presentation of few interoperability problems (SOAP 1.1)
   - Presentation of interoperability
   - SOAP encoding and message types
   - WS-I tool

4. Divers
   - Next session 06/07/09 at 2.00 pm
   - Structure of final report
   - The template for the flyer will be arranged by Mr. Joye
   - Decision: Automatic generation of WSDL file has no high priority

JOB FOR THE NEXT WEEK:

1. Finish the presented documents and tasks

2. Document discussed concepts

3. Divers

David Schneider, 30/06/09

KEYMILE access to the world	Manage telecommunication using Web Services Protocol 06.07.2009 / 14h00 – 15h30	Seite / Page 1/1 Next session date **Over**

Students:	**EIF Profs**:	**Copy to** :
Thierry KIKI ✓	Philippe JOYE ✓	
David SCHNEIDER ✓	François BUNTSCHU ✓	
	KEYMILE :	N. MAYENCOURT (Expert) ✓
	Daniel GACHET (Mandatory) ✓	Marcel BELLORINI ✓

LIST OF POINTS:

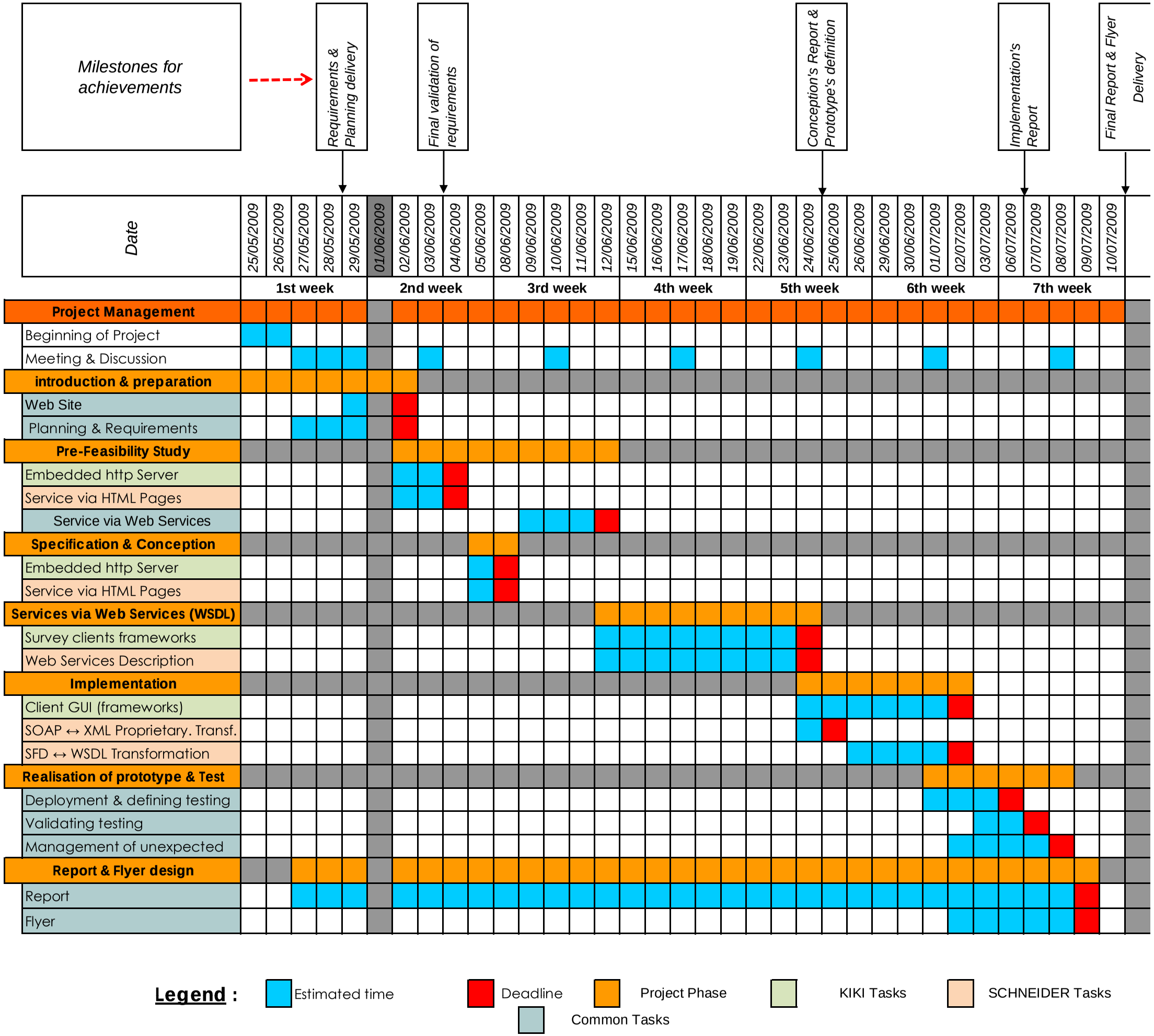
1. Summary of last session
   - Difference between SOAP RPC/Document
   - WS-I Profile description

2. Presentation of the performed work
   - Framework Installation
   - Problems uncured
   - WS-Concept
   - Discussion on flyer

3. Test
   - Generate manually SOAP Request
   - WSDL validation
   - Test Definition

Annexe: Planing

Diploma Projet : Manage telecommunication equipement via Web Services



Legend : Estimated time · Deadline · Project Phase · KIKI Tasks · SCHNEIDER Tasks · Common Tasks

Project Planning **Version 1.0** **Thierry KIKI & David SCHNEIDER**

Annexe: HTML Service

Table of Contents

1 Definitions

MO:	Managed Object
MOM:	Managed Object Model
moType:	Managed Object Type
MF:	Management Function
ADF:	AccesPoint (MO) – definition file
MCST:	MileGate Configuration Software Tool
KOAP:	KEYMILE Object Access Protocol
SOAP:	Simple Object Access Protocol (W3C recommendation)
WebServices:	W3C recommendation
GUI:	Graphical User Interface

2 Initiation

2.1 Project Objective

Find a good way to generate on the fly HTML pages within the MilGate which is providing a web browser access.

2.2 Background

It would be interesting to offer a possibility to display and modify the configuration of the MileGate network device for humans. The most simple and standardized way is to provide the access via a web browser as a lot of other network devices as routers, modems, acces points or switches do.

Our only interface to access the data or configuration parameters is the MileGate Object Model with its proprietary communication protocol.

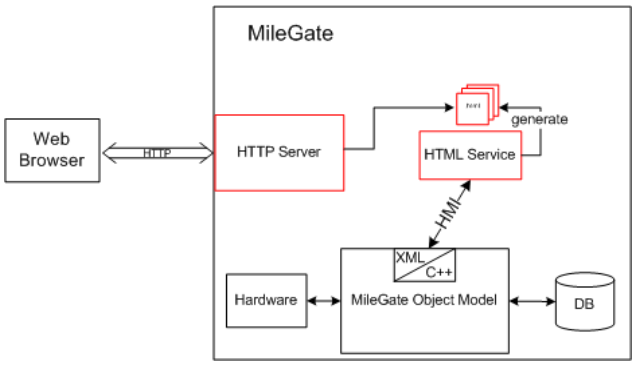


Illustration 1: System structure

For further treatment of the data for the presentation layer, we need to know the overall structure of the configuration (possible parameters) which needs to be parsed from an XML Schema, the ADF (proprietary AccessPoint Definition File) or in the future from the description of our Web Service (WSDL File).

2.3 Scope

2.3.1 Business functions

The aim is to analyze the feasibility of a service on the MileGate which creates HTML pages on the fly (run-time). It must be possible to change the configuration of the MileGate via an web browser.

It is not possible and not wished to to have the complete information in the memory because we would create redundancy which is complex to to manage.

It is imaginable to save the navigation structure on the system but all the data will be requested on use.

The service must be adaptable with a modular structure. Also the presentation layer and the logic must be separated strictly.

2.3.2 Project interfaces

The service will run on the MileGate core card (limitations will be discussed later) and is accessing the management interface. Information about the complete system are published in the document “User Guide – MileGate & MCST”.

For the implementation in C++, the document “C++ Programming Style Guidelines, Common Part” and “C++ Programming Practice Guidelines, Common Part” need to be respected.

2.3.3 Required analysis

Embedded System limits

- use of memory
- performance of system

Actual management system

- functions
- operational implementation

Constraints for MileGate

Identification of problems for implementation

2.4 Project constraints

2.4.1 Project dates

This task of the project is initially limited at 5 workdays. It is possible to resume some parts at an advanced project state.

2.4.2 Interproject dependencies

The task does not dependent on further work of our project but we can eventually identify common problems. The survey of the actual system and interface will help to understand the functioning and simplify future workings.

3 Analysis

3.1 Embedded System limits

3.1.1 Memory limitation

The memory of the MileGate is limited and has to be used with fully aware. The program itself need to be adapted to the MileGate coding rules.

If its necessary to add images or other graphical elements, they could be loaded over the Internet. HTML is pure text and does not use lot of memory.

Core Card: 128MB / 256MB of RAM
128MB Flash Memory (no hard disk drive)

3.1.2 Performance limitation

The actual management system (MCST) generates a lot of request towards the management interface. Amelioration is possible but not vital.

Performance limitations rather have to be considered at the implementation of the HTML Service due to the generation of the HTML files and its storage uses much more system resources.

CPU: PowerPC 603E (~400MHz)

3.2 Communication with the MileGate

3.2.1 Describe the Client-Server system used to communication with the MileGate

The communication with the management interface uses a proprietary XML protocol named KOAP which is transported over a proprietary message transport protocol (replaced in future by SOAP sent with HTTP/HTTPS)

It is a matter of a simple request-response system. The client is allowed to send request and the server (MileGate management interface) returns a response with the an indication whether the request was successful or had an error.

The KOAP protocol additionally offers the possibilities to send attachments.

All the services handling the configuration must access this management interface.

3.2.2 Describe the format of the requests and responses

The following paragraph shows how the transmitted message should look like.

The actual management interface accepts request which looks as followed:

```
<?xml version="1.0" encoding="utf-8"?>
<request version="1" seq="1" destAddr="/unit-1/port-1">
        <mdomain id="main">
                <operation seq="1" name="setLabel">
                        <Label>
                                <user>User1</user>
                                <service>Service1</service>
                                <description>Description1</description>
                        </Label>
                </operation>
        </mdomain>
</request>
```

Illustration 2: KOAP request

The request addresses the Management Object Type (MO Type) "/unit-1/port-1" and the Management Function (MF) "main". The called function is named setLabel and requires the shown XML formatting.

For the response we observe the response on the function getLabel because the function used just before won't deliver any content. The response looks as followed:

```
<?xml version="1.0" encoding="utf-8"?>
<response version="1" seq="1" destAddr="/unit-1/port-1">
        <mdomain id="main">
                <operation seq="1" name="getLabel">
                        <execution status="success"/>
                        <Label>
                                <user>User1</user>
                                <service>Service1</service>
                                <description>Description1</description>
                        </Label>
                </operation>
        </mdomain>
</request>
```

Illustration 3: KOAP response

The requests has the same parameters as the request. Additionally the tag <execution> with the parameter status="success" has been added into the tag <operation>. An unsuccessful response would contain the execution parameter status="proc_error".

Within the <operation> tag, the values just send before in the setLabel function were returned.

3.3 Object Model and the actual management system (MCST)

3.3.1 Analise the structure of the Object Model

An introduction to the MileGate Object Model MOM can be found in the document "Introduction to the MileGate XML Management Interface" under "Minimal introduction to the MOM".

The tree is built by Managed Objects (MO) in a hierarchical model.

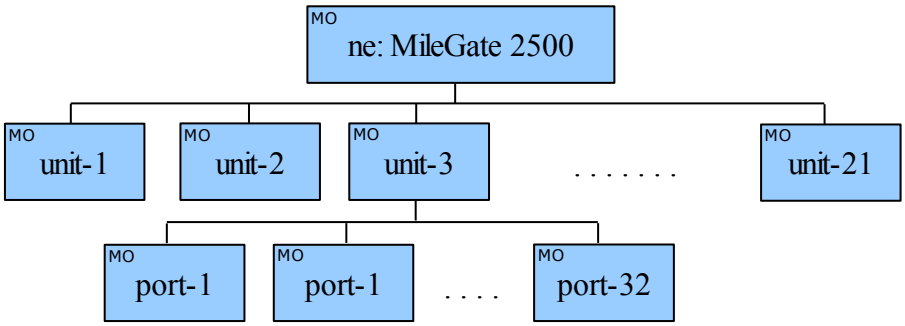


Illustration 4: MileGate Object Model structure

Each MO has its proper set of Management Functions (MF).

MF of root node: Main, Configuration, Fault Management, Status
MF of unit node: Main, Configuration, Fault Management, Status
MF of type port: Main, Configuration, Fault Management, Performance Management, Status

The complete structure of the **MileGate 2500** MF's is represented as annexe.

3.3.2 Analise the functional design

The first operation the MCST needs to know what type of equipment we are about to connect. Therefor a **Discover** message has to be sent to the root node's main management function.

```
<info>
   <moType>ne.milegate.2500</moType>
   <adfReference>keyne_r2e05pr_ws</adfReference>
   <addressFragment>ne</addressFragment>
   <moName>MileGate 2500</moName>
   <assignedMoName/>
   <state>ok</state>
   <adminState>na</adminState>
   <label>
      <user/>
      <service/>
      <description/>
   </label>
   <uuid>
      <id/>
   </uuid>
   <maxApAlarmSeverity>minor</maxApAlarmSeverity>
   <maxPropagatedAlarmSeverity>warning</maxPropagatedAlarmSeverity>
   <lastConfigChangedSeqNr>0</lastConfigChangedSeqNr>
   <lastSavedSeqNr>0</lastSavedSeqNr>
   <configChangedByLoad>false</configChangedByLoad>
   <hasChildren>true</hasChildren>
   <koapVersion>1</koapVersion>
   <eqpStatus>unprotected</eqpStatus>
</info>
.
<ChildList>

   .
   .
```

```
            <info>
               <moType>unit.holder.extended</moType>
               <adfReference>keyne_r2e05pr_ws</adfReference>
               <addressFragment>/unit-11</addressFragment>
               <moName>COGE1 R1D</moName>
               <assignedMoName>keyne_r2e05pr_ws</assignedMoName>
               <state>plugged</state>
               <adminState>down</adminState>
               <label>
                  <user/>
                  <service/>
                  <description/>
               </label>
               <uuid>
                  <id/>
               </uuid>
               <maxApAlarmSeverity>warning</maxApAlarmSeverity>
               <maxPropagatedAlarmSeverity>cleared</maxPropagatedAlarmSeverity>
               <lastConfigChangedSeqNr>0</lastConfigChangedSeqNr>
               <lastSavedSeqNr>0</lastSavedSeqNr>
               <configChangedByLoad>true</configChangedByLoad>
               <hasChildren>true</hasChildren>
               <koapVersion>2</koapVersion>
               <eqpStatus>unprotected</eqpStatus>
            </info>
            .
```

Illustration 5: Response of Discover

Additionally to the information about the equipment, the discover request provides a list of its children.

The complete structure of the Managed Object Type (MOType) **ne.milegate.2500** can be looked up in the AccessPoint Description File (ADF).

The GUI of the actual configuration tool MCST is generated automatically by parsing this ADF file.

```
      <mf name="main">
         <group name="general" cli="General" gui="General">
            <property name="Label" cli="Labels" gui="Labels">
               <struct name="Label" cli="Labels" gui="Labels">
                  <value name="user" type="string" range="63" gui="Label 1"/>
                  <value name="service" type="string" range="63" gui="Label
2"/>
                  <value name="description" type="string" range="127"
                     gui="Description"/>
```

```
            </struct>
         </property>
         <property name="AlarmSeverity" gui="Alarm Status">
            <struct name="AlarmSeverity" gui="Alarm Status">
               <enum name="maxAlarmSeverity" gui="Highest Alarm Severity">
                  <symbol name="cleared"gui="Cleared"/>
                  <symbol name="indeterminate" gui="Indeterminate"/>
                  <symbol name="warning" gui="Warning"/>
                  <symbol name="minor" gui="Minor"/>
                  <symbol name="major" gui="Major"/>
                  <symbol name="critical" gui="Critical"/>
               </enum>
               <enum name="maxPropagatedAlarmSeverity" gui="Highest
Propagated Alarm Severity">
                  <symbol name="cleared" gui="Cleared" helpText=""/>
                  <symbol name="indeterminate" gui="Indeterminate"/>
                  <symbol name="warning" gui="Warning"/>
                  <symbol name="minor" gui="Minor"/>
                  <symbol name="major" gui="Major"/>
                  <symbol name="critical" gui="Critical"/>
               </enum>
            </struct>
         </property>
      </group>
```

Illustration 6: ADF structure

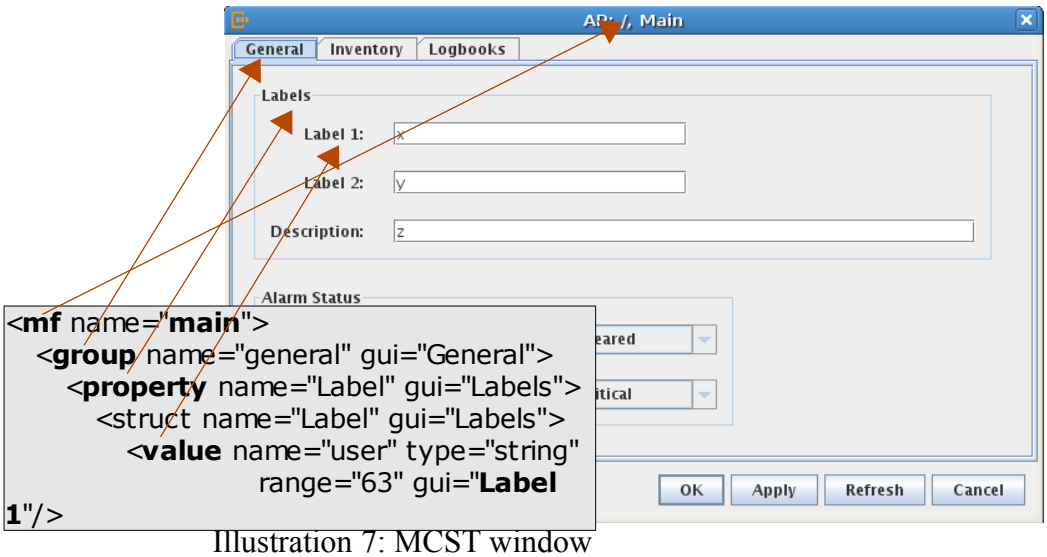


Illustration 7: MCST window

The structure is complete now, but the values are still missing. To get the values we need to send the proprietary KOAP message we mentioned before to the desired node.

To make the link between the ADF file, the KOAP message and the automatical generated GUI we represent once again the response for the getLabel function. The request was directed to the Management Function (MF) **main** with the property **Label** and the action **GET**.

```
<Label>
   <user>x</user>                  // ADF: <value name="user" gui="Label 1"/>
   <service>y</service>            // ADF: <value name="service" gui="Label 2"/>
   <description>z</description>    // ADF: <value name="description" gui="Label 3"/>
</Label>
```

Illustration 8: MileGate Object Model structure

3.3.3 Analise the dynamic adaptation mechanism

The MCST loads the complete tree of Managed Objects at the opening of the Application or if the user clicks on the refresh button.

If the user presses the refresh button, the entire management function MF (for example: main or configuration) will be regenerated with all its KOAP requests. This technique has the disadvantage that if we change just the timezone (MF: configuration), 41 KOAP requests need to be generated, sent and answered.

Those requests are generated very fast and it does not use much CPU usage to response them. But if we need to generate 15 new HTML pages (assumption that the structure of the GUI won't be changed) for each changing in this configuration management function, we waste lot of resources.

At the implementation of the HTML Service, we have to consider that we just send KOAP messages for the parameters which have changed.

4 Feasibility studies

4.1 Identify problems for implementation

Problem	Description	Mitigation
Parsing HTML	It is difficult to extract information from HTML pages as it doesn't have a well defined structure.	The parsing of XML is much easier in C++. It could be a good solution to use XHTML instead of HTML.
Memory limitation for complete database. We can either create a DB for the service or always request the wanted parameters.	DB: + must get just modified parameter for regeneration - memory Direct output: + simpler to implementation - content of entire page must be requested on each modification	Creation of a DB probably won't be necessary for this implementation. We create additional problems caused by duplicating the data. Likewise is the implemented SAX parser on MileGate not optimal for the creation of a DB. In my opinion it's better to keep the number of request as small as possible.
Menu structure	The menu is complex but needs to be well arranged at the same time.	A good technique to use would be a solution based a tree menu (example JavaScript) for the navigation within the nodes and kind of pop-up menu for the management functions.
Refresh of navigation menu on insertion of new unit	The menu must be updated (rewrite HTML page) if a new unit is inserted. On the browser it can be reloaded automatically with a refresh timing.	We need to detect the low-level interrupt! To find the accurate method the survey of the MCST will be helpful.
File transfer on HTML	The actual system initiates file transfer with a tag and adds the file just behind. This is possible due to the protocol is no standardized.	Here we have to study how to use HTTP/Put in C++
Acknowledge on modification	If the user modifies a parameter, he needs to be sure that the operation was successful.	We can not send messages to the user with HTML (HTTP Server is between service and client). The only possibility is to print error messages on the HTML page

		which will be visible on the next reload.
Config of multiple Managed Objects (MO's)	The MCST GUI offers the possibility of configuring multiple MO's with one action.	This is difficult to implement in HTML, the task needs further studies.
Connection Manager (access the node)	The MCST GUI offers a connection manager which is user dependent.	The connection parameters of the users can not be managed trough the server. It is possible to use cookies to save connection parameters on the users web browser.
Customizing the GUI / Custom toolbar	A helpful add-on of the MCST is the customizable interface.	It will be very challenging to implement a customizable HTML page. The feasibility and its advantages should be studied in a further task. A custom toolbar is rather conceivable. It must also be saved on the client machine with a technologie such as cookies.
Printing option / Table CSV export	The MCST GUI offers a printing option and table export possibilities for spreadsheet programs.	Printing in HTML is obtainable with a well formated page or a additional stylesheet. The export possibility is more difficult and probably not supported in HTML. The CSV files may need to be generated within our HTML Service instead.

4.2 Description of a possible implementation

We want a product which is as modular and adaptable as possible. To achieve this we certainly need a strict separation between logic and presentation.

logic:

- contains parsing of structure and management of data (get/set via MOM interface)
- parsing of the object model
- interface between HTML and KOAP
- detection of changes

- new card: → parse
- modified config: → what to add/modify

presentation:

- contains presentation of data and generation/modification of navigation
- the data need to be represented in function of its usability/type
- the navigation need to be generated automatically
- Plug/unplug must modify the navigation according to the logic

5 Recommendation for Implementation

This topic contains our recommendation for the implementation of the HTML Service and an example user interface. The recommendations are based on the prior studies and converge in the basic structure towards the actual management system. This was necessary because no deep study on the structure was performed and with this, no change can be recommended.

5.1 Use Case diagram

The following diagram is a first approach to describe a possible functionality of the system.

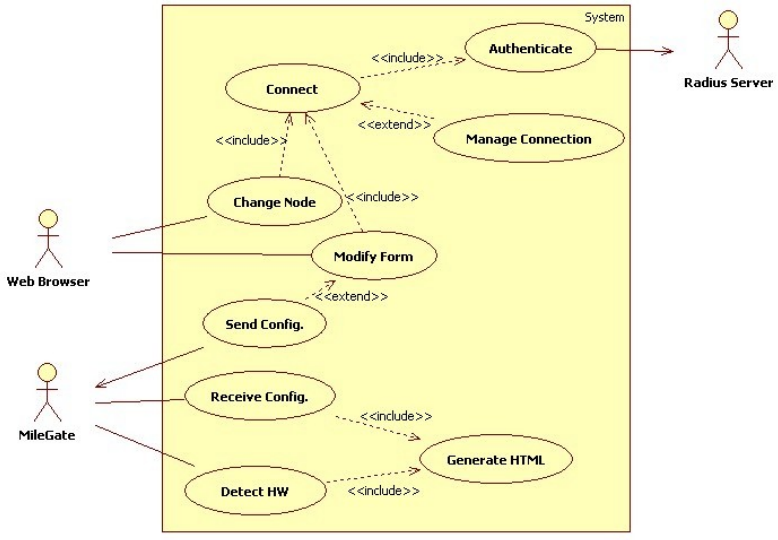


Illustration 9: Use Case diagram

5.2 Sequence Diagram

With the sequence diagram we like to show the sequential interactions and exchange of messages.

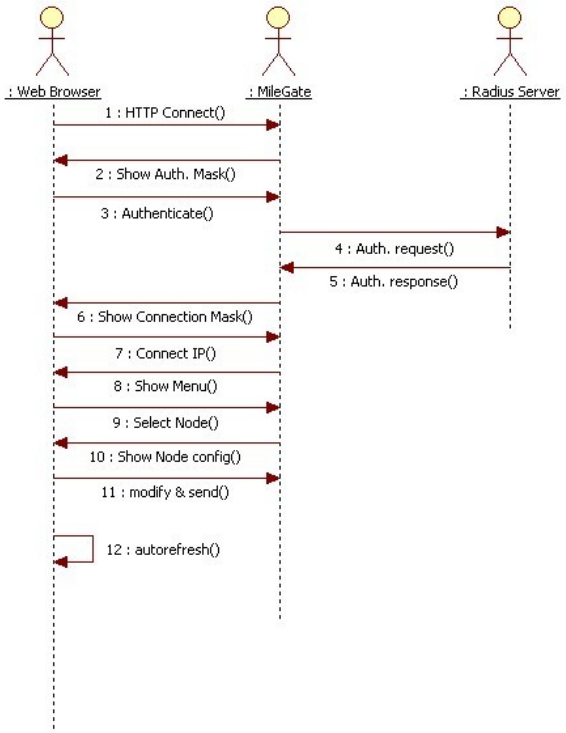
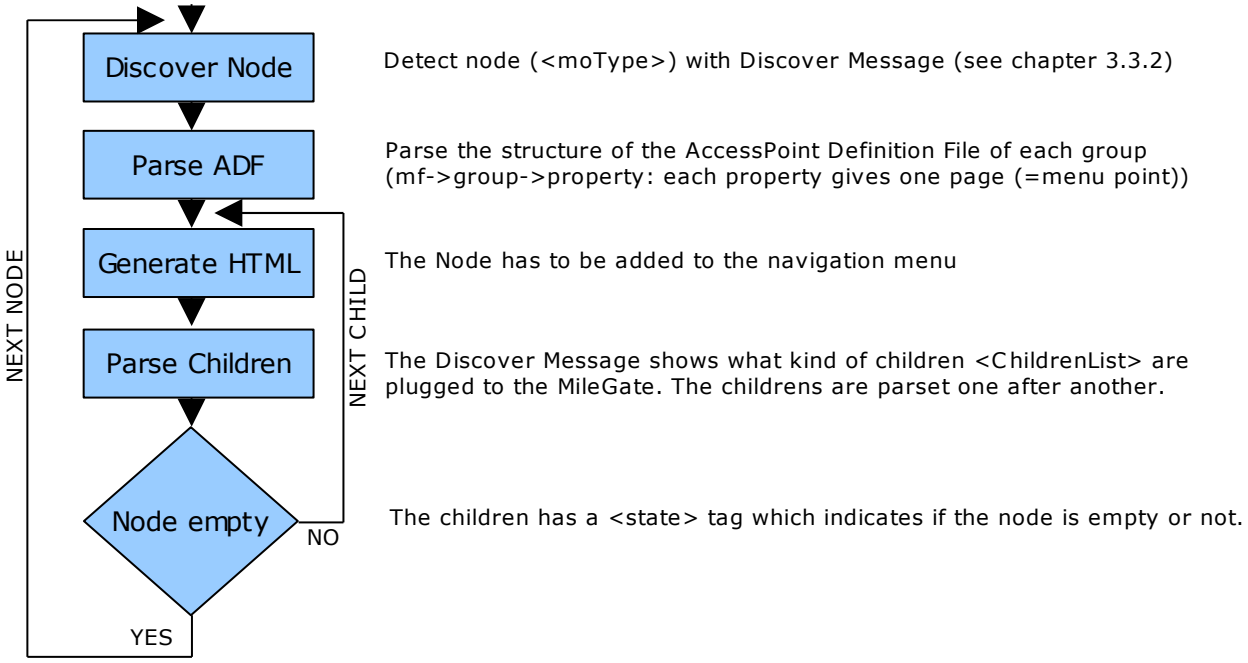


Illustration 10: Sequence Diagram

5.3 Operation of the HTML Service

At the initiation of the HTML service, the entire navigation structure has to be generated. The result of this will be accessible by the client after the step 7 of the Sequence Diagram. The connection itself does not evoke the initiation of the service, the structure needs to existing already at this point of time.

The following points visualize the basic functionality of the service and describe how the service can figure out the structure of the node.



It has to be said that the parsing of objects has to be recursive which is not represented in this flowchart.

5.4 GUI Prototype

The menu is the most important part of the website because it defines the way we can navigate trough the sites and with this the ease of use. Basically we have the root node with its units and ports. Further a technique need to be evaluated to add maximal five additional menus to access the further navigation structure (Main, Configuration Management, Fault Management, Performance Management and Status) of each node. Possibilities are a second navigation frame or a pop-up accessible by the right mouse button.

The following illustration provides a GUI prototype with a second navigation frame and pull down menus.

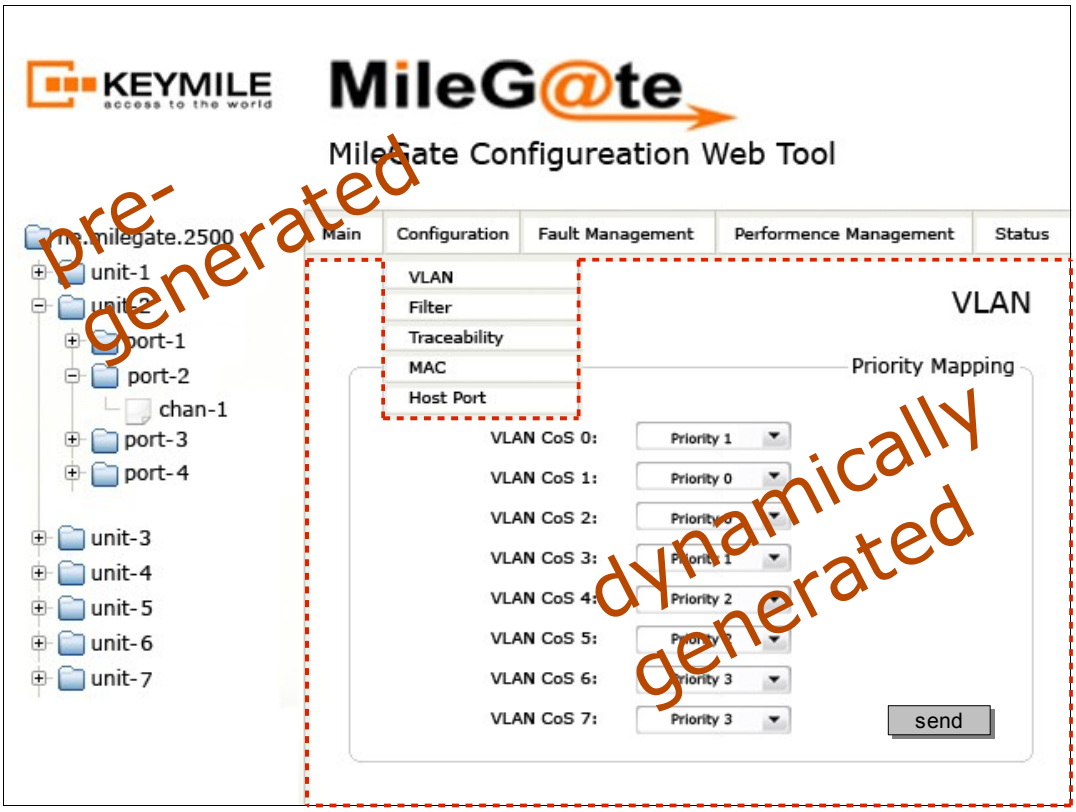


Illustration 11: GUI prototype

The website needs to be built with frames. That way we can use one single menu (left) on every other page. In function of the selection on the left side, the top menu and its menu points (top) need to change. As described before, the structure of this menu is defined in the ADF file for any possible kind of node.

If the user clicks on a navigation point, the real task for the service has to be performed. As we do not want to save the pages with the parameters anywere, we have to generate the entire content (exclusive of menus) at this point of time.

Request of content:

1. The possible form fields, check boxes, combo boxes, tables or buttons of one content frame is defined in the ADF file.
2. Transformation between ADF XML and HTML/XHTML has to be performed
3. To get the values we have to send KOAP messages with indication which parameters we would like (in the example case it would be:
request destAddr=”/”, mdomain id=”cfgm”, operation name=”getPriorityMapping”)
4. The service needs to merge the XHTML code and the parameters
5. Finally the XHTML has to be saved on memory
6. With a proper configuration of the HTTP Server, the file is now accessible by the user

5.5 Generation of HTML files

The generation of the HTML files is similar to the transformation already used for the JAVA client MCST. This is just the case if we decide to keep the actual structure described in the ADF file (see "Analise the functional design").

ADF structure: <group><property><struct> or <group><property><table>

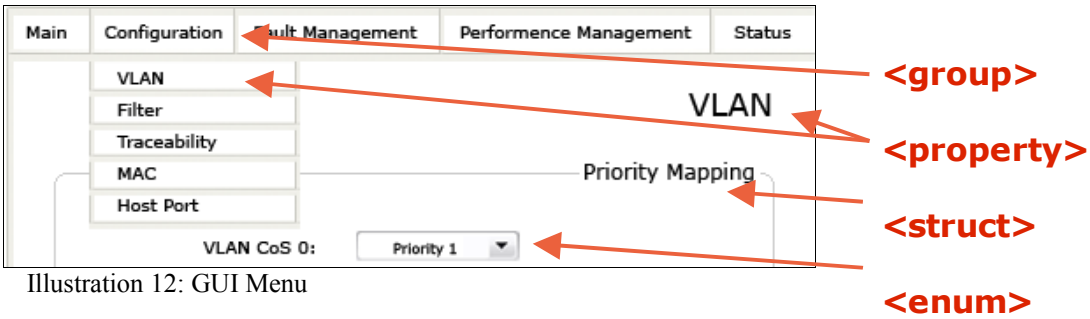


Illustration 12: GUI Menu

<group> → MENU Points

<property> → Page

<struct>/<table> → Content Element (multiple per page)

Within the <struct>, the tags could be transformed as followed:

<enum> : Combo Box

<value> : Text/Form Field

The transformation of the <table> tag is more complex.

5.6 Reaction on modification

The system needs to react to modification automatically. Modifications are possible on different interfaces such as CLI, MCST, syslog, SNMP and of course the web interface for this service.

The MileGate generates notification on a change of the configuration. Those notifications need to be captured by our service and must generate the new navigation automatically. This needs to be considered at the conception of the navigation structure. Those changes also complicates the automatic generation of the navigation structure.

5.7 Reaction on new Hardware

If a new unit is added or removed, the node needs to be added in the navigation menu and of course also deleted.

On addition of a unit a similar mechanism as the one described in the topic “Operation of the HTML service” has to be performed starting at the added unit instead of the root node.

5.8 Problems

Additional to the identification of the problems in the point “Feasibility studies” we list here some very important points for the implementation of the service.

Error Handling:

To announce errors to the user, we can just use the output of the HTML page. It is possible to generate error pages or to add the error message at any place of the page. We need to define what will be shown during the generation process to give the best feedback to the user.

Concurrent Problems:

The handling of concurrent access need to be checked to guarantee the functionality. In some cases, the access or the files have to be locked for secondary users.

Refresh Problems:

Automatic refresh of the HTML page with a refresh delay could cause some problems. We also have to pay attention that the caching mechanism of the browser/website is configured well. We need a very quick refresh time to not confuse/irritate the user.

Performance Problems:

We saw in the analysis that the embedded system has some limitations such as the performance. To avoid performance problems, proper testing is necessary.

6 Conclusion

A service which generates HTML pages on the MileGate is feasible.

The aim (advantage compared to MCST) and the wanted functions of such a service need to be planed and analyzed carefully. Main advantage is that a client does not need to install anything. It is also imaginable that browsers on mobile devices can be used.

The diagrams and process descriptions have no deep complexity and need to be expanded. Primary aim of those presentations was to provide an overview of the conceived approach.

At my point of view, a customizable user interface or a change of the look-and-feel could bring some advantages for the use of the interface.

The required time to realize this project is very difficult to estimate at the actual state because it depends heavily on the desired functionalities.

The survey of the actual management tool (MCST) and its implementation helped a lot and completed the introduction into the very complex MileGate system. We feel certain that this analysis will facilitate future tasks and helps if such a service will be designed.

7 Annexes

7.1 Revision history

Revision			Short description of the modification	Prepared by	Checked by	Approved
Doc ID	Version	Date				
HTML	1	06.09.09	First Version	DSCHN		
HTML	1.1	15/06/09	Modification after discussion on project session. (Major changes on chapters 2.3.1, 5, 5.3, 5.4, 5.6)	DSCHN		

7.2 References

- User Guide – MileGate & MCST
- C++ Programming Style Guidelines, Common Part (KEYMILE Confidential)
- C++ Programming Practice Guidelines, Common Part (KEYMILE Confidential)
- Extensible HyperText Markup Language http://www.w3.org/TR/xhtml1/

7.3 Structure of the MileGate 2500 Management Functions

The Structure can vary on different MileGate models, Units and Port types!

RootNode (MileGate 2500)

- Main
  - General
    - Lables: — Text fields
      - AlarmStatus: — Combo boxes
  - Investory
    - Equipment Inventory: — Text fields
  - Equipment
    - Database Compatibility List: — Text fields
      - Database Compatibility Function: — Table
  - Logbooks: — Buttons
- Configuration
  - Date And Time
    - SNTP Client
      - Operation Mode: — Combo boxes
      - Primary Server: — Text field
      - Secondary Server: — Text field
      - Polling Interval: — Text field
      - Broadcast Delay: — Text field
    - Time Zone: — Combo/Checkbox
  - IPSec
    - IPSec Parameters: — Checkbox
      - IPSec Policy: — Table
      - Internet Key Exchange: — Table
      - Security Methods: — Table
  - QoS
    - 802.1p Priority Mapping
      - 802.1p Priority – To – Queue Mapping: — Table
  - SNMP
    - General: — Button
      - SNMP Support: — Checkbox
      - SNMP Parameters: — Text fields
  - Syslog Destinations
    - Destination 1
      - Destination Configuration: — Text fields
        - Source: — Table

.

.

Destination 10:

Destination Configuration: Text fields

Source: Table

Syslog Source

Syslog Source List: Table

Temperature Limits

Exceed: Text fields

Warning: Text fields

ESO

ESO-1 Clock Sources: Check boxes

ESO-2: Check boxes

PETS

PDH Clock Sources: Combo boxes

PETS Clock Priority: Combo boxes

PETS Extraction: Combo box

Modify Password: Button

Configuration Management: Buttons

Configuration Status: Combo/Table

Management Interface

Management Interface: Text fields

Management VLAN: Text field

Packet

Bridge: Combo boxes

Traceability: Text field

PPPoA: Text fields

Session Management

Retry Time: Text field

Sessionmanager Session Timeout: Text field

Authentication Management Interface: Check boxes

RADIUS Default Userclass: Combo box

Radius Client

Radius Common Parameters: Check/ text fields

Primary Radius Server: Check/ text fields

Alternate Radius Server: Check/ text fields

Fault Management

Active Failures

Failure: Table

Status: Button

Alarm Status: Table

Configuration

Alarm Configuration: Table

Status

Radius Client
- Primary Radius Server Status: Combo/ text fields
- Alternate Radiius Server Status: Combo/ text fields

IPSec
- SA Status: Table

Redundancy: Buttons
- NE Configuration Status
  - Overall Configuration Status: Check box
  - Detailed Configuration Status: Combo/ Check box
- Core Unit Roles: Button/ text field
- Core Unit Status: Check boxes

Temperature: Button
- Current: Text field
- Max.: Text field
- Min.: Text field

ESO
- ESO-1: Combo box
- ESO-2: Combo box

PETS: Button
- PETS Clock Sourcfes: Check Table
- PETS Status: Combo box

Date And Time
- Time: Button
  - Summary: Combo/ text fields
- SNTP Client
  - Primary Server Status: Combo box
  - Secondary Server Status: Combo box
  - Last Response Time: Text fields
  - Last Jump Time: Text fields
  - Last Adjustment Time: Text fields

Management Interface: Button
- SSH Fingerprint: Text fields

Session Management
- Session: Table

Unit (SUAD1)

Main

General

Lables: Text fields

AlarmStatus: Combo boxes

Equipment: Buttons

Assignment Status: Combo/ text field

Equipment Status: Combo/ text fields

System Compatibility List: Text field

System Compatibility Functions: Table

Database Compatibility List: Text fields

Database Compatibility Functions: Table

Inventory

Equipment Inventory: Text fields

Internal Logbooks: Button

Cpuload: Buttons

File Access

Transfer: Button

Delete: Buttons

Software

Software On Element Manager: Table

Software On Unit: Table

Disk Space: Text fields

Configuration: Combo/ text fields

Configuration

VLAN

Priority Mapping: Combo boxes

Filter

Security Filtering: Combo boxes

Traceability

Logon Options

DHCP: Combo box

Aging: Combo/ text field

PPPoE: Combo box

MAC

MAC Options: Text field

Host Port

Policer Profile: Button/ Combo box

Fault Management

Status: Button

Alarm Status: Table

Configuration: Table

Commands: Buttons

Port (ADSL on SUAD1)

- Main
  - General
    - Lables: Text boxes
    - Alarm Status: Combo boxes
  - Admin And Oper Status
    - Administrative Status: Combo box
    - Operational Staus: Combo box
- Configuration
  - Multicast: Button/ Check boxes
    - Maximum Number Of Multicast Streams
    - Group Management
      - Multicast Access Profile 1: Button/ Checkbox
      - Multicast Access Profile 2: Button/ Checkbox
      - Multicast Access Profile 3: Button/ Checkbox
  - Traceability
    - Agent Remote ID: Text field
  - Security
    - Maximum Number Of Addresses for TLS: Text field
    - Maximum Number Of MAC Addresses For Nto1: Text field
  - Access Control
    - Classification Key: Combo box
  - Rate Limiter
    - Rate Limiter
      - Upstream Profile: Button/Combo/Check
      - Downstream Profile: Button/Combo/Check
  - QoS
    - Scheduling Profile: Button/Combo
  - Profile
    - Port Profile: Button/Combo
- Fault Management
  - Status: Button
    - Alarm Status: Table
  - Configuration
    - Alarm Configuration: Check Table/Button
- Performance Management
  - Performance Monitoring: Check boxes/Buttons
    - UserCounter: Table
    - History 15min Table
    - History 24h Table
- Status
  - General

DSL Status:	Combo box
Attainable Rate:	Text fields
Output Power:	Text fields
Band Table:	Table
Statistics	
Port Counters:	Text fields
Policing Counter:	Text fields
MAC Access Dynamic List	
Unicast List:	Table
MAC Forwarding List:	Button
MAC Forwarding List:	Table
QoS	
Weighted Fair Queues:	Text field
Multicast	
Stream:	Button
Active Streams:	Table/ Text field
VLAN	
Attached VLANs:	Table/ Text field
Defects	
Defects	
Central Office:	Check boxes
CPE:	Check boxes
Line Inventory	
DSL Vendor	
Central Office:	Text fields
CPE:	Text fields
Maintenance	
DSL Operation Status:	Combo box
RFI Bands	
Notch Table Downstream:	Table

Annexe: WSDL

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="mob_mainbase"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:mob_mainbase_xml="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/management"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  targetNamespace="http://www.keymile.com/milegate/ws/mob_mainbase_xml">


<!-- === DOCUMENTATION === -->

  <documentation>
    -textual description of Web Service
    -further infos for the use of this service or interface
    -contact person
  </documentation>

<!-- === TYPES === -->

  <types>
    <xs:schema xmlns="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:_0="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://www.keymile.com/milegate/ws/mob_mainbase_xml">
      <!-- FAULT TYPE -->
      <xs:element name="Fault" type="Fault__Type"/>
      <xs:complexType name="Fault__Type">
        <xs:sequence>
          <xs:element name="Code">
            <xs:complexType>
              <xs:sequence minOccurs="0" maxOccurs="1">
                <xs:element name="Value"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Reason">
            <xs:complexType>
              <xs:sequence minOccurs="0" maxOccurs="1">
                <xs:element name="Text"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
      <!--MAINBASE TYPES-->
      <xs:element name="Info2" type="Info2__Type"/>
      <xs:complexType name="Info2__Type">
        <xs:sequence>
          <xs:element name="moType">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="63"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="adfReference">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:maxLength value="63"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="addressFragment">
            <xs:simpleType>
              <xs:restriction base="xs:string">
```

```
            <xs:maxLength value="80"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="moName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="assignedMoName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="state" type="mob_mainbase_xml:EquipmentState__Type"/>
      <xs:element name="adminState" type="AdminStateReadOnly__Type"/>
      <xs:element name="label" type="Label__Type"/>
      <xs:element name="uuid" type="Uuid__Type"/>
      <xs:element name="maxApAlarmSeverity" type="severity__Type"/>
      <xs:element name="maxPropagatedAlarmSeverity" type="severity__Type"/>
      <xs:element name="lastConfigChangedSeqNr">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="lastSavedSeqNr">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="configChangedByLoad">
        <xs:simpleType>
          <xs:restriction base="xs:boolean"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="hasChildren">
        <xs:simpleType>
          <xs:restriction base="xs:boolean"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="koapVersion">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="eqpStatus" type="EqpStatus__Type"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Discover" type="Discover__Type"/>
  <xs:complexType name="Discover__Type">
    <xs:sequence>
      <xs:element name="info" type="Info2__Type"/>
      <xs:element name="ChildList">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="info" type="Info2__Type"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="severity" type="severity__Type"/>
  <xs:simpleType name="severity__Type">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="notification"/>
      <xs:enumeration value="cleared"/>
      <xs:enumeration value="indeterminate"/>
      <xs:enumeration value="warning"/>
      <xs:enumeration value="minor"/>
      <xs:enumeration value="major"/>
      <xs:enumeration value="critical"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="restricted_severity" type="restricted_severity__Type"/>
  <xs:simpleType name="restricted_severity__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="notification"/>
      <xs:enumeration value="cleared"/>
      <xs:enumeration value="indeterminate"/>
      <xs:enumeration value="warning"/>
      <xs:enumeration value="minor"/>
      <xs:enumeration value="major"/>
      <xs:enumeration value="critical"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="executionStatus" type="executionStatus__Type"/>
  <xs:simpleType name="executionStatus__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="success"/>
      <xs:enumeration value="error_no_such_ap"/>
      <xs:enumeration value="error_no_such_md"/>
      <xs:enumeration value="error_no_such_it"/>
      <xs:enumeration value="error_no_such_operation"/>
      <xs:enumeration value="error_wrong_param"/>
      <xs:enumeration value="error_out_of_range"/>
      <xs:enumeration value="error_operation_not_applicable"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="AdminState" type="AdminState__Type"/>
  <xs:simpleType name="AdminState__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="up"/>
      <xs:enumeration value="down"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="AdminStateReadOnly" type="AdminStateReadOnly__Type"/>
  <xs:simpleType name="AdminStateReadOnly__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="up"/>
      <xs:enumeration value="down"/>
      <xs:enumeration value="testing"/>
      <xs:enumeration value="na"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="OperState" type="OperState__Type"/>
  <xs:simpleType name="OperState__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="up"/>
      <xs:enumeration value="down"/>
      <xs:enumeration value="testing"/>
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="dormant"/>
      <xs:enumeration value="notPresent"/>
      <xs:enumeration value="lowerLayerDown"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="UserClass" type="UserClass__Type"/>
  <xs:simpleType name="UserClass__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="information"/>
```

```
      <xs:enumeration value="maintenance"/>
      <xs:enumeration value="manager"/>
      <xs:enumeration value="support"/>
      <xs:enumeration value="debug"/>
      <xs:enumeration value="sessionmanager"/>
      <xs:enumeration value="softwaremanager"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="EqpStatus" type="EqpStatus__Type"/>
  <xs:simpleType name="EqpStatus__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unprotected"/>
      <xs:enumeration value="standby"/>
      <xs:enumeration value="active"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Label" type="Label__Type"/>
  <xs:complexType name="Label__Type">
    <xs:sequence>
      <xs:element name="user">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="service">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="description">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="127"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Uuid" type="Uuid__Type"/>
  <xs:complexType name="Uuid__Type">
    <xs:sequence>
      <xs:element name="id">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="LastConfigChanged" type="LastConfigChanged__Type"/>
  <xs:complexType name="LastConfigChanged__Type">
    <xs:sequence>
      <xs:element name="address">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="80"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="lastConfigChangedSeqNr">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
```

```
        </xs:simpleType>
      </xs:element>
      <xs:element name="lastSavedSeqNr">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="configChangedByLoad">
        <xs:simpleType>
          <xs:restriction base="xs:boolean"/>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="configChanged" type="configChanged__Type"/>
  <xs:complexType name="configChanged__Type">
    <xs:sequence>
      <xs:element name="list">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="LastConfigChanged" type="LastConfigChanged__Type"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="configChangedIn" type="configChangedIn__Type"/>
  <xs:complexType name="configChangedIn__Type">
    <xs:sequence>
      <xs:element name="start">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="AlarmSeverity" type="AlarmSeverity__Type"/>
  <xs:complexType name="AlarmSeverity__Type">
    <xs:sequence>
      <xs:element name="maxAlarmSeverity" type="severity__Type"/>
      <xs:element name="maxPropagatedAlarmSeverity" type="severity__Type"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="InfraVersion" type="InfraVersion__Type"/>
  <xs:simpleType name="InfraVersion__Type">
    <xs:restriction base="xs:string">
      <xs:maxLength value="10"/>
    </xs:restriction>
  </xs:simpleType>

  <!--MAINEQUIPMENT TYPES-->
  <xs:simpleType name="EquipAssignState__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="unassigned"/>
      <xs:enumeration value="assigned"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="EquipmentState" type="EquipmentState__Type"/>
  <xs:simpleType name="EquipmentState__Type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="empty"/>
      <xs:enumeration value="plugged"/>
      <xs:enumeration value="ok"/>
      <xs:enumeration value="mismatch"/>
      <xs:enumeration value="failed"/>
```

```
      <xs:enumeration value="na"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="EquipAssignStatus" type="EquipAssignStatus__Type"/>
  <xs:complexType name="EquipAssignStatus__Type">
    <xs:sequence>
      <xs:element name="state" type="EquipAssignState__Type"/>
      <xs:element name="assignedHw">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="EquipmentStatus" type="EquipmentStatus__Type"/>
  <xs:complexType name="EquipmentStatus__Type">
    <xs:sequence>
      <xs:element name="state" type="EquipmentState__Type"/>
      <xs:element name="currentHwRevision">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="currentSwRevision">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="currentSerialNumber">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="currentManufName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="currentModelName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="EquipmentInventory" type="EquipmentInventory__Type"/>
  <xs:complexType name="EquipmentInventory__Type">
    <xs:sequence>
      <xs:element name="symbol">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
```

```
<xs:element name="shortText">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="63"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="boardId">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedShort"/>
  </xs:simpleType>
</xs:element>
<xs:element name="hardwareKey">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedShort"/>
  </xs:simpleType>
</xs:element>
<xs:element name="manufacturerId">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="manufacturerSerialNumber">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="manufacturerPartNumber">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="manufacturerBuildState">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="supplierPartNumber">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="supplierBuildState">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="customerId">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="customerProductId">
```

```
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="bootloaderRevision">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="31"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="processor">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="MainEquipAssign_DB" type="MainEquipAssign_DB__Type"/>
  <xs:complexType name="MainEquipAssign_DB__Type">
    <xs:sequence>
      <xs:element name="state" type="EquipAssignState__Type"/>
      <xs:element name="assignedHw">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="63"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="LBookContents" type="LBookContents__Type"/>
  <xs:complexType name="LBookContents__Type">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="LBookEntry">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Entry">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:maxLength value="400"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadTaskHistory" type="CpuloadTaskHistory__Type"/>
  <xs:complexType name="CpuloadTaskHistory__Type">
    <xs:sequence>
      <xs:element name="taskName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="32"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="taskId">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="12"/>
```

```
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="taskPrio">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="12"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period1">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period2">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period3">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period4">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period5">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period6">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period7">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period8">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="period9">
```

```
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="period10">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="period11">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="period12">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadHistoryArray" type="CpuloadHistoryArray__Type"/>
  <xs:complexType name="CpuloadHistoryArray__Type">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="CpuloadTaskHistory__Type" type="CpuloadTaskHistory__Type"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadHistory" type="CpuloadHistory__Type"/>
  <xs:complexType name="CpuloadHistory__Type">
    <xs:sequence>
      <xs:element name="measurementStart">
        <xs:simpleType>
          <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="measurementEnd">
        <xs:simpleType>
          <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="array" type="CpuloadHistoryArray__Type"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadTaskHistoryRaw" type="CpuloadTaskHistoryRaw__Type"/>
  <xs:complexType name="CpuloadTaskHistoryRaw__Type">
    <xs:sequence>
      <xs:element name="taskName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="32"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="taskId">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="taskPrio">
```

```
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="history">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="value">
              <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadHistoryRaw" type="CpuloadHistoryRaw__Type"/>
  <xs:complexType name="CpuloadHistoryRaw__Type">
    <xs:sequence>
      <xs:element name="clkPerSec">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="periodDuration">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="measurementStart">
        <xs:simpleType>
          <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="measurementEnd">
        <xs:simpleType>
          <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="startPeriod">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="nbrPeriods">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="task">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="CpuloadTaskHistoryRaw__Type" type="CpuloadTaskHistoryRaw__Type"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CpuloadHistoryRawInput" type="CpuloadHistoryRawInput__Type"/>
  <xs:complexType name="CpuloadHistoryRawInput__Type">
    <xs:sequence>
      <xs:element name="startPeriod">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>
```

```
            <xs:element name="maxNbrPeriods">
              <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
              </xs:simpleType>
            </xs:element>
            <xs:element name="minPercent">
              <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:element name="DbgFtDirectory" type="DbgFtDirectory__Type"/>
        <xs:simpleType name="DbgFtDirectory__Type">
          <xs:restriction base="xs:string">
            <xs:maxLength value="127"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:element name="DbgFtFilePath" type="DbgFtFilePath__Type"/>
        <xs:simpleType name="DbgFtFilePath__Type">
          <xs:restriction base="xs:string">
            <xs:maxLength value="127"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:element name="CheckUnitAssignParam" type="CheckUnitAssignParam__Type"/>
        <xs:complexType name="CheckUnitAssignParam__Type">
          <xs:sequence>
            <xs:element name="isProtectingUnit">
              <xs:simpleType>
                <xs:restriction base="xs:boolean"/>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:schema>
    </types>

<!-- === MESSAGES === -->

    <!-- WS-Management headers  -->
    <message name="ResourceURIMessage">
      <part name="Header" element="wsman:ResourceURI"/>
    </message>
    <message name="SelectorSetMessage">
      <part name="Header"  element="wsman:SelectorSet"/>
    </message>

    <!-- WS-Addressing headers  -->
    <message name="ToMessage">
      <part name="Header"  element="wsa:To"/>
    </message>
    <message name="ReplyToMessage">
      <part name="Header"  element="wsa:ReplyTo"/>
    </message>
    <message name="ActionMessage">
      <part name="Header"  element="wsa:Action"/>
    </message>
    <message name="MessageIDMessage">
      <part name="Header"  element="wsa:MessageID"/>
    </message>

    <!-- bodys  -->
    <!-- FAULT MESSAGE -->
    <message name="errorMessage">
      <part name="Error" element="mob_mainbase_xml:Fault"/>
    </message>
    <!-- MAINBASE MESSAGES -->
```

```
  <message name="EmptyMessage">
  </message>
  <message name="Info2__Message">
    <part name="Body" element="mob_mainbase_xml:Info2"/>
  </message>
  <message name="Discover__Message">
    <part name="Body" element="mob_mainbase_xml:Discover"/>
  </message>
  <message name="severity__Message">
    <part name="Body" element="mob_mainbase_xml:severity"/>
  </message>
  <message name="restricted_severity__Message">
    <part name="Body" element="mob_mainbase_xml:restricted_severity"/>
  </message>
  <message name="executionStatus__Message">
    <part name="Body" element="mob_mainbase_xml:executionStatus"/>
  </message>
  <message name="AdminState__Message">
    <part name="Body" element="mob_mainbase_xml:AdminState"/>
  </message>
  <message name="AdminStateReadOnly__Message">
    <part name="Body" element="mob_mainbase_xml:AdminStateReadOnly"/>
  </message>
  <message name="OperState__Message">
    <part name="Body" element="mob_mainbase_xml:OperState"/>
  </message>
  <message name="UserClass__Message">
    <part name="Body" element="mob_mainbase_xml:UserClass"/>
  </message>
  <message name="EqpStatus__Message">
    <part name="Body" element="mob_mainbase_xml:EqpStatus"/>
  </message>
  <!-- link between Message(used in Interface <operation> Set&GetLabel) <element
name="Label">(structure) -->
  <message name="Label__Message">
    <part name="Body" element="mob_mainbase_xml:Label"/>
  </message>
  <message name="Uuid__Message">
    <part name="Body" element="mob_mainbase_xml:Uuid"/>
  </message>
  <message name="LastConfigChanged__Message">
    <part name="Body" element="mob_mainbase_xml:LastConfigChanged"/>
  </message>
  <message name="configChanged__Message">
    <part name="Body" element="mob_mainbase_xml:configChanged"/>
  </message>
  <message name="configChangedIn__Message">
    <part name="Body" element="mob_mainbase_xml:configChangedIn"/>
  </message>
  <message name="AlarmSeverity__Message">
    <part name="Body" element="mob_mainbase_xml:AlarmSeverity"/>
  </message>
  <message name="InfraVersion__Message">
    <part name="Body" element="mob_mainbase_xml:InfraVersion"/>
  </message>
  <!-- MAINEQUIPEMENT MESSAGES -->
  <message name="EquipAssignState__Message">
    <part name="Body" element="mob_mainbase_xml:EquipAssignState"/>
  </message>
  <message name="EquipmentState__Message">
    <part name="Body" element="mob_mainbase_xml:EquipmentState"/>
  </message>
  <message name="EquipAssignStatus__Message">
    <part name="Body" element="mob_mainbase_xml:EquipAssignStatus"/>
  </message>
  <message name="EquipmentStatus__Message">
    <part name="Body" element="mob_mainbase_xml:EquipmentStatus"/>
```

```
  </message>
  <message name="EquipmentInventory__Message">
    <part name="Body" element="mob_mainbase_xml:EquipmentInventory"/>
  </message>
  <message name="MainEquipAssign_DB__Message">
    <part name="Body" element="mob_mainbase_xml:MainEquipAssign_DB"/>
  </message>
  <message name="LBookSelection__Message">
    <part name="Body" element="mob_mainbase_xml:LBookSelection"/>
  </message>
  <message name="LBookContents__Message">
    <part name="Body" element="mob_mainbase_xml:LBookContents"/>
  </message>
  <message name="CpuloadTaskHistory__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadTaskHistory"/>
  </message>
  <message name="CpuloadHistoryArray__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadHistoryArray"/>
  </message>
  <message name="CpuloadHistory__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadHistory"/>
  </message>
  <message name="CpuloadTaskHistoryRaw__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadTaskHistoryRaw"/>
  </message>
  <message name="CpuloadHistoryRaw__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadHistoryRaw"/>
  </message>
  <message name="CpuloadHistoryRawInput__Message">
    <part name="Body" element="mob_mainbase_xml:CpuloadHistoryRawInput"/>
  </message>
  <message name="DbgFtDirectory__Message">
    <part name="Body" element="mob_mainbase_xml:DbgFtDirectory"/>
  </message>
  <message name="DbgFtFilePath__Message">
    <part name="Body" element="mob_mainbase_xml:DbgFtFilePath"/>
  </message>
  <message name="CheckUnitAssignParam__Message">
    <part name="Body" element="mob_mainbase_xml:CheckUnitAssignParam"/>
  </message>

<!-- === PORT TYPES === -->

  <portType name="main_base__PortType">
    <!-- MAINBASE PORT TYPES -->
    <operation name="GetLabel__Operation">
      <input message="mob_mainbase_xml:EmptyMessage"/>
      <output message="mob_mainbase_xml:Label__Message"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="SetLabel__Operation">
      <input message="mob_mainbase_xml:Label__Message"/>
      <output message="mob_mainbase_xml:Label__Message"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="GetAlarmSeverity__Operation">
      <input message="mob_mainbase_xml:EmptyMessage"/>
      <output message="mob_mainbase_xml:AlarmSeverity__Message"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="GetInfo2__Operation">
      <input message="mob_mainbase_xml:EmptyMessage"/>
      <output message="mob_mainbase_xml:Info2__Message"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="GetDiscover__Operation">
      <input message="mob_mainbase_xml:EmptyMessage"/>
```

```
    <output message="mob_mainbase_xml:Discover__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="GetInfraVersionTest__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:InfraVersion__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="GetUuid__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:Uuid__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="SetUuid__Operation">
    <input message="mob_mainbase_xml:Uuid__Message"/>
    <output message="mob_mainbase_xml:Uuid__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <!-- MAINEQUIPMENT PORT TYPES -->
  <operation name="GetEquipAssignStatus__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EquipAssignStatus__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="GetEquipmentStatus__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EquipmentStatus__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="Assign__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EmptyMessage"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="Unassign__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EmptyMessage"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="RestartUnit__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EmptyMessage"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="StopBoot__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EmptyMessage"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="CheckUnitAssign__Operation">
    <input message="mob_mainbase_xml:CheckUnitAssignParam__Message"/>
    <output message="mob_mainbase_xml:EmptyMessage"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="GetInventory__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:EquipmentInventory__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="doGetLBook__Operation">
    <input message="mob_mainbase_xml:LBookSelection__Message"/>
    <output message="mob_mainbase_xml:LBookContents__Message"/>
    <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
  </operation>
  <operation name="Cpuload__Operation">
    <input message="mob_mainbase_xml:EmptyMessage"/>
    <output message="mob_mainbase_xml:CpuloadHistory__Message"/>
```

```
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="CpuloadRaw__Operation">
      <input message="mob_mainbase_xml:CpuloadHistoryRawInput__Message"/>
      <output message="mob_mainbase_xml:CpuloadHistoryRaw__Message"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="DbgFtDir__Operation">
      <input message="mob_mainbase_xml:DbgFtDirectory__Message"/>
      <output message="mob_mainbase_xml:EmptyMessage"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="DbgFtDeleteDir__Operation">
      <input message="mob_mainbase_xml:DbgFtDirectory__Message"/>
      <output message="mob_mainbase_xml:EmptyMessage"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
    <operation name="DbgFtDeleteFile__Operation">
      <input message="mob_mainbase_xml:DbgFtFilePath__Message"/>
      <output message="mob_mainbase_xml:EmptyMessage"/>
      <fault name="Fault" message="mob_mainbase_xml:errorMessage"/>
    </operation>
  </portType>

<!-- === BINDING === -->

  <binding name="main_base__Interface" type="mob_mainbase_xml:main_base__PortType">
    <soapbind:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <!-- MAINBASE BINDING -->
    <operation name="GetLabel__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal">
        </soapbind:header>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </output>
       <fault name="Fault">
         <soapbind:fault use="literal" name="Fault" />
       </fault>
    </operation>
    <operation name="SetLabel__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Set"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
```

```
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetAlarmSeverity__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetInfo2__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetDiscover__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetInfraVersionTest__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
```

```
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetUuid__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="SetUuid__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Set"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <!-- MAINEQUIPMENT BINDING -->
    <operation name="GetEquipAssignStatus__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
```

```
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetEquipmentStatus__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="Assign__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="Unassign__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="RestartUnit__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
```

```
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="StopBoot__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="CheckUnitAssign__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="GetInventory__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
```

```
      <soapbind:body use="literal"/>
    </output>
    <fault name="Fault">
      <soapbind:fault use="literal" name="Fault" />
    </fault>
  </operation>
  <operation name="doGetLBook__Operation">
    <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
    <input>
      <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
      <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
      <soapbind:body use="literal"/>
    </input>
    <output>
      <soapbind:body use="literal"/>
    </output>
    <fault name="Fault">
      <soapbind:fault use="literal" name="Fault" />
    </fault>
  </operation>
  <operation name="Cpuload__Operation">
    <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
    <input>
      <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
      <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
      <soapbind:body use="literal"/>
    </input>
    <output>
      <soapbind:body use="literal"/>
    </output>
    <fault name="Fault">
      <soapbind:fault use="literal" name="Fault" />
    </fault>
  </operation>
  <operation name="CpuloadRaw__Operation">
    <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
    <input>
      <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
      <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
      <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
      <soapbind:body use="literal"/>
    </input>
    <output>
      <soapbind:body use="literal"/>
    </output>
    <fault name="Fault">
      <soapbind:fault use="literal" name="Fault" />
    </fault>
  </operation>
  <operation name="DbgFtDir__Operation">
    <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
    <input>
      <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
```

```
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="DbgFtDeleteDir__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
    <operation name="DbgFtDeleteFile__Operation">
      <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get"/>
      <input>
        <soapbind:header message="mob_mainbase_xml:ResourceURIMessage" part="Header" use="literal"
/>
        <soapbind:header message="mob_mainbase_xml:SelectorSetMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ReplyToMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:ActionMessage" part="Header" use="literal" />
        <soapbind:header message="mob_mainbase_xml:MessageIDMessage" part="Header" use="literal" />
        <soapbind:body use="literal"/>
      </input>
      <output>
        <soapbind:body use="literal"/>
      </output>
      <fault name="Fault">
        <soapbind:fault use="literal" name="Fault" />
      </fault>
    </operation>
  </binding>


<!-- === SERVICE === -->

  <service name="LabelService">
    <port name="LabelPort" binding="mob_mainbase_xml:main_base__Interface">
      <soapbind:address location="http://localhost:9357/wsman/" /><!-- HTTP Destination Addresse, Just for
testing, to be overwrite in frame work -->
      <wsa:EndpointReference name="LabelEPR"
        xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl">
        <wsa:Address>http://localhost:9357/wsman/</wsa:Address>
        <wsa:ReferenceParameters>
```

```
            <wsman:SelectorSet>
              <wsman:Selector name="mf">main</wsman:Selector>
              <wsman:Selector name="property">/Label</wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
          </wsa:ReferenceParameters>
        </wsa:EndpointReference>
      </port>
      <port name="DiscoverPort" binding="mob_mainbase_xml:main_base__Interface">
        <soapbind:address location="http://localhost:9357/wsman/" /><!-- HTTP Destination Addresse, Just for
testing, to be overwrite in frame work -->
        <wsa:EndpointReference name="DiscoverEPR"
          xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl">
          <wsa:Address>http://localhost:9357/wsman/</wsa:Address>
          <wsa:ReferenceParameters>
            <wsman:SelectorSet>
              <wsman:Selector name="mf">main</wsman:Selector>
              <wsman:Selector name="property">/Discover</wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
          </wsa:ReferenceParameters>
        </wsa:EndpointReference>
      </port>
      <port name="Info2Port" binding="mob_mainbase_xml:main_base__Interface">
        <soapbind:address location="http://localhost:9357/wsman/" /><!-- HTTP Destination Addresse, Just for
testing, to be overwrite in frame work -->
        <wsa:EndpointReference name="DiscoverEPR"
          xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl">
          <wsa:Address>http://localhost:9357/wsman/</wsa:Address>
          <wsa:ReferenceParameters>
            <wsman:SelectorSet>
              <wsman:Selector name="mf">main</wsman:Selector>
              <wsman:Selector name="property">/Info2</wsman:Selector>
            </wsman:SelectorSet>
            <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
          </wsa:ReferenceParameters>
        </wsa:EndpointReference>
      </port>
    </service>
</definitions>
```

Annexe: XSLT

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This XSLT stylesheet generates SADF, a special ADF-like file for the simulator. The generation process is similar to
the ADF generation.
-->
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sfd="http://www.keymile.com/sfd"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/management"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  exclude-result-prefixes="#all">

  <xsl:import href="fpmd_generator.xslt"/>
  <xsl:output method="xml" indent="yes"/>
  <xsl:strip-space elements="*"/>
  <xsl:param name="name"/>
  <xsl:param name="sfds"/>
  <xsl:variable name="sfd">
    <xsl:for-each-group select="tokenize($sfds, '\s+')" group-by=".">
      <xsl:if test="string-length(.) > 0">
        <!-- add base uri information for error messages -->
        <xsl:variable name="uri" select="translate(., '\\', '/')"/>
        <sfd uri="{$uri}">
          <xsl:copy-of select="document($uri)/sfd/*"/>
        </sfd>
      </xsl:if>
    </xsl:for-each-group>
  </xsl:variable>

  <xsl:variable name="wsdlNS">http://schemas.xmlsoap.org/wsdl/</xsl:variable>
  <xsl:variable name="xsdNS">http://www.w3.org/2001/XMLSchema</xsl:variable>
  <xsl:variable name="xsiNS">http://www.w3.org/2001/XMLSchema-instance</xsl:variable>
  <xsl:variable name="soapNS">http://schemas.xmlsoap.org/wsdl/soap12/</xsl:variable>
  <xsl:variable name="wsmanNS">http://schemas.xmlsoap.org/ws/2005/06/management</xsl:variable>
  <xsl:variable name="wsaNS">http://schemas.xmlsoap.org/ws/2004/08/addressing</xsl:variable>

  <!-- the keymile base namespace for milegate web-services -->
  <xsl:variable name="mgNS">http://www.keymile.com/milegate/ws/</xsl:variable>
  <xsl:variable name="mgwsNS">http://www.keymile.com/milegate/ws/ws-common</xsl:variable>

  <xsl:template match="/" >
    <sadf name="{substring-before($name, '_')}" revision="{substring-after($name, '_')}">

      <!-- remove this, used for checking what is in $sfd. contents are copied into
           the SADF file -->
      <xsl:copy-of select="$sfd"/>

      <!-- write a file that displays the SFD hierachies, specially to detect
           cyclic includes -->

      <xsl:result-document href="wsdl/!sfd_include-hierarchy.xml">
        <include-hierarchy>
        <xsl:for-each select="$sfd/sfd">
          <xsl:sort select="upper-case(header/@name)"/>
          <SFD-FILE name="{header/@name}">
            <xsl:apply-templates select=".//include" mode="show-includes"/>
          </SFD-FILE>
        </xsl:for-each>
        </include-hierarchy>
      </xsl:result-document>

      <xsl:for-each select="$sfd/sfd">
        <!-- for each SFD file ... -->
        <xsl:variable name="tokenizedUri" select="tokenize(@uri, '/|\.')"/>
```

```
<xsl:variable name="fileName" select="$tokenizedUri[count($tokenizedUri)-1]"/>

<xsl:result-document href="wsdl/{$fileName}.wsdl">

  <!-- get the namespace for this file -->
  <xsl:variable name="NS" select="concat($mgNS, header/@name)"/>
  <!-- generate the <wsdl:definitions> element -->
  <xsl:element namespace="{$wsdlNS}" name="definitions">
    <!-- namespace for this file -->
    <xsl:namespace name="{header/@name}" select="$NS"/>
    <!-- static namespaces -->
    <xsl:namespace name="xsi"  select="$xsiNS"/>
    <xsl:namespace name="mgws" select="$mgwsNS"/>
    <xsl:namespace name="soapbind" select="$soapNS"/>
    <xsl:namespace name="wsman" select="$wsmanNS"/>
    <xsl:namespace name="wsa" select="$wsaNS"/>

    <!-- namespaces for the imports -->
    <xsl:apply-templates select="externals/include" mode="namespaces"/>
    <!-- targetNamespace -->
    <xsl:attribute name="targetNamespace" select="$NS"/>

    <!-- generate the imports -->
    <xsl:text>&#xA;&#xA;</xsl:text>
    <xsl:comment> === IMPORTS === </xsl:comment>
    <xsl:text>&#xA;&#xA;</xsl:text>
    <xsl:element namespace="{$wsdlNS}" name="import">
      <xsl:attribute name="namespace" select="$mgwsNS"/>
      <xsl:attribute name="location"  select="'ws-common.wsdl'"/>
    </xsl:element>
    <xsl:apply-templates select="externals/include" mode="imports"/>

    <!-- generate the types -->
    <xsl:text>&#xA;&#xA;</xsl:text>
    <xsl:comment> === TYPES === </xsl:comment>
    <xsl:text>&#xA;&#xA;</xsl:text>
    <xsl:element namespace="{$wsdlNS}" name="types">
      <xsl:element namespace="{$xsdNS}" name="schema">
        <xsl:namespace name="" select="$NS"/>
        <xsl:namespace name="xs"   select="$xsdNS"/>
        <xsl:attribute name="elementFormDefault" select="'qualified'"/>
        <xsl:attribute name="targetNamespace" select="$NS"/>
        <xs:element name="Fault" type="Fault__Type"/>                    <!-- New fault element -->
        <xs:complexType name="Fault__Type">
          <xs:sequence>
            <xs:element name="Code">
              <xs:complexType>
                <xs:sequence minOccurs="0" maxOccurs="1">
                  <xs:element name="Value"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="Reason">
              <xs:complexType>
                <xs:sequence minOccurs="0" maxOccurs="1">
                  <xs:element name="Text"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <!-- apply for all global types... -->
        <xsl:apply-templates select="types/*" mode="globalTypes"/>
      </xsl:element>
    </xsl:element>

    <!-- generate the messages -->
```

```
        <xsl:text>&#xA;&#xA;</xsl:text>
        <xsl:comment> === MESSAGES === </xsl:comment>
        <xsl:text>&#xA;&#xA;</xsl:text>
        <!-- generate the messages headers →                    <!-- New SOAP message headers-->
        <xsl:comment> WS-Management headers  </xsl:comment>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'ResourceURIMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsman:ResourceURI'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'SelectorSetMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsman:SelectorSet'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
         <xsl:text>&#xA;&#xA;</xsl:text>
        <xsl:comment> WS-Addressing headers  </xsl:comment>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'ToMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsa:To'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'ReplyToMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsa:ReplyTo'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'ActionMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsa:Action'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'MessageIDMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Header'"/>
            <xsl:attribute name="element" select="'wsa:MessageID'"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:text>&#xA;&#xA;</xsl:text>

        <!-- generate the fault message -->
        <xsl:comment> FAULT MESSAGE </xsl:comment>                    <!-- New Fault Message-->
        <xsl:element namespace="{$wsdlNS}" name="message">
          <xsl:attribute name="name" select="'errorMessage'"/>
          <xsl:element namespace="{$wsdlNS}" name="part">
            <xsl:attribute name="name" select="'Error'"/>
            <xsl:attribute name="element" select="sfd:addOwnNsIfMissing(./header, 'Fault')"></xsl:attribute>
          </xsl:element>
        </xsl:element>
        <xsl:text>&#xA;&#xA;</xsl:text>
        <xsl:comment> BODY's </xsl:comment>
        <!-- generate the messages, one per global type -->
        <xsl:apply-templates select="types/*" mode="messages"/>

        <!-- generate the port types and operations.
            for now each sfd:mf is mapped to a port type -->
```

```
            <!-- first generate it into a temporary tree so that we can re-use
                 the results for the binding -->
            <xsl:variable name="PortTypes">
              <xsl:text>&#xA;&#xA;</xsl:text>
              <xsl:comment> === PORT TYPES === </xsl:comment>
              <xsl:text>&#xA;&#xA;</xsl:text>
              <!-- generate a port type for each mf but only if it contains operations -->
              <xsl:apply-templates select="management-functions/mf[.//property|.//command]" mode="port-types"/
>
            </xsl:variable>
            <xsl:copy-of select="$PortTypes"/>

            <!-- generate the binding.
                 for now each sfd:mf is mapped to a port type is bound to an interface -->
            <xsl:text>&#xA;&#xA;</xsl:text>
            <xsl:comment> === BINDING === </xsl:comment>
            <xsl:text>&#xA;&#xA;</xsl:text>
            <xsl:apply-templates select="$PortTypes/*" mode="binding">
              <xsl:with-param name="NS" select="header/@name"/>
            </xsl:apply-templates>

          </xsl:element>
        </xsl:result-document>
      </xsl:for-each>
    </sadf>
  </xsl:template>

<!-- templates to display include hierarchies ===
========================================== -->

  <xsl:template match="include" mode="show-includes">
    <xsl:param name="level" select="1"/>
    <xsl:variable name="importedSfdFileName" select="concat('/', .)"/>
    <xsl:variable name="importedWsdlFilename" select="concat(substring-before(., '.'), '.wsdl')"/>
    <xsl:variable name="importedSfdNode"  select="$sfd/sfd[ends-with(@uri, $importedSfdFileName)]"/>
    <xsl:variable name="importedHeaderName" select="$importedSfdNode/header/@name"/>
    <xsl:if test="$level != 100">
      <file level="{$level}" name="{$importedHeaderName}">
        <xsl:apply-templates select="$importedSfdNode//include" mode="#current">
          <xsl:with-param name="level" select="$level+1"/>
        </xsl:apply-templates>
      </file>
    </xsl:if>
  </xsl:template>

<!-- templates for the imported files
======================================================== -->

  <xsl:template match="include" mode="imports">
    <!-- fileName shall hold something like '/syslog_xml.sfd', the name of the imported file -->
    <xsl:variable name="importedSfdFileName" select="concat('/', .)"/>
    <xsl:variable name="importedWsdlFilename" select="concat(substring-before(., '.'), '.wsdl')"/>
    <xsl:variable name="importedSfdNode"  select="$sfd/sfd[ends-with(@uri, $importedSfdFileName)]"/>
    <xsl:variable name="importedHeaderName" select="$importedSfdNode/header/@name"/>
    <!-- generate the <wsdl:import> element -->
    <xsl:element namespace="{$wsdlNS}" name="import">
      <xsl:attribute name="namespace" select="concat($mgNS, $importedHeaderName)"/>
      <xsl:attribute name="location"  select="$importedWsdlFilename"/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="include" mode="namespaces">
    <xsl:variable name="importedSfdFileName" select="concat('/', .)"/>
    <xsl:variable name="importedSfdNode"     select="$sfd/sfd[ends-with(@uri, $importedSfdFileName)]"/>
    <xsl:variable name="importedHeaderName"  select="$importedSfdNode/header/@name"/>
    <!-- generate the namespaces -->
```

```
    <xsl:namespace name="{$importedHeaderName}" select="concat($mgNS, $importedHeaderName)"/>
  </xsl:template>

<!-- templates for the types
================================================================== -->

  <xsl:template match="ipaddress" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|
2[0-4][0-9]|25[0-5])){{3}}){{0,1}}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xsl:template>

  <xsl:template match="ipaddress" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:string">
        <xs:pattern value="(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|
2[0-4][0-9]|25[0-5])){{3}}){{0,1}}"/>
      </xs:restriction>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="macaddress" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:simpleType>
        <xs:restriction base="xs:hexBinary">
          <xs:length value="6"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xsl:template>

  <xsl:template match="macaddress" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:hexBinary">
        <xs:length value="6"/>
      </xs:restriction>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="moAddress" mode="nestedTypes">
    <xs:element name="{@name}" type="xs:string"/>
  </xsl:template>

  <xsl:template match="moAddress" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="string" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="{@max-length}"/>
          <xsl:if test="encoding">
            <xs:pattern value="\p{IsBasicLatin}*"/>
          </xsl:if>
        </xs:restriction>
      </xs:simpleType>
```

```
    </xs:element>
  </xsl:template>

  <xsl:template match="string" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:string">
        <xs:maxLength value="{@max-length}"/>
        <xsl:if test="encoding">
          <xs:pattern value="\p{IsBasicLatin}*"/>
        </xsl:if>
      </xs:restriction>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="hexBinary" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:simpleType>
        <xs:restriction base="xs:hexBinary">
          <xs:maxLength value="{@max-length}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xsl:template>

  <xsl:template match="hexBinary" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:hexBinary">
        <xs:maxLength value="{@max-length}"/>
      </xs:restriction>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="boolean|date|time|dateTime|byte|double|float|int|long|short|unsignedByte|unsignedInt|
unsignedLong|unsignedShort" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:simpleType>
        <xs:restriction base="xs:{name()}">
          <xsl:if test="@min">
            <xs:minInclusive value="{@min}"/>
          </xsl:if>
          <xsl:if test="@max">
            <xs:maxInclusive value="{@max}"/>
          </xsl:if>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xsl:template>

  <xsl:template match="boolean|date|time|dateTime|byte|double|float|int|long|short|unsignedByte|unsignedInt|
unsignedLong|unsignedShort" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:simpleType name="{sfd:getTypeName(@name)}">
      <xs:restriction base="xs:{name()}">
        <xsl:if test="@min">
          <xs:minInclusive value="{@min}"/>
        </xsl:if>
        <xsl:if test="@max">
          <xs:maxInclusive value="{@max}"/>
        </xsl:if>
      </xs:restriction>
    </xs:simpleType>
  </xsl:template>

  <xsl:template match="xmlFragment" mode="nestedTypes">
    <xs:any maxOccurs="1" minOccurs="1" processContents="skip"/>
```

```
</xsl:template>

<xsl:template match="xmlFragment" mode="globalTypes">
  <!-- do not generate anything for global fragements -->
</xsl:template>

<xsl:template match="bitset" mode="nestedTypes">
  <xs:element name="{@name}">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="(0|1){{{@bits}}}"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xsl:template>

<xsl:template match="bitset" mode="globalTypes">
  <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
  <xs:simpleType name="{sfd:getTypeName(@name)}">
    <xs:restriction base="xs:string">
      <xs:pattern value="(0|1){{{@bits}}}"/>
    </xs:restriction>
  </xs:simpleType>
</xsl:template>

<xsl:template match="enum" mode="nestedTypes">
  <xs:element name="{@name}">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xsl:for-each select="symbol">
          <xs:enumeration value="{@name}"/>
        </xsl:for-each>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xsl:template>

<xsl:template match="enum" mode="globalTypes">
  <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
  <xs:simpleType name="{sfd:getTypeName(@name)}">
    <xs:restriction base="xs:string">
      <xsl:for-each select="symbol">
        <xs:enumeration value="{@name}"/>
      </xsl:for-each>
    </xs:restriction>
  </xs:simpleType>
</xsl:template>

<xsl:template match="choice|profile" mode="nestedTypes">
  <xs:element name="{@name}" type="xs:string"/>
</xsl:template>

<xsl:template match="choice|profile" mode="globalTypes">
</xsl:template>

<xsl:template match="description" mode="nestedTypes"/>

<xsl:template match="description" mode="globalTypes"/>

<xsl:template match="struct" mode="nestedTypes">
  <xs:element name="{@name}">
    <xs:complexType>
      <xs:sequence>
        <xsl:apply-templates select="*" mode="nestedTypes"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  </xsl:template>

  <xsl:template match="struct" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:complexType name="{sfd:getTypeName(@name)}">
      <xs:sequence>
        <xsl:apply-templates select="*" mode="nestedTypes"/>
      </xs:sequence>
    </xs:complexType>
  </xsl:template>

  <xsl:template match="array|table" mode="nestedTypes">
    <xs:element name="{@name}">
      <xs:complexType>
        <xs:sequence minOccurs="{if (@size) then @size else 0}" maxOccurs="{if (@size) then @size else if
(@maximal-size) then @maximal-size else 'unbounded'}">
          <xsl:apply-templates select="*" mode="nestedTypes"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xsl:template>

  <xsl:template match="array|table" mode="globalTypes">
    <xs:element name="{@name}" type="{sfd:getTypeName(@name)}"/>
    <xs:complexType name="{sfd:getTypeName(@name)}">
      <xs:sequence minOccurs="{if (@size) then @size else 0}" maxOccurs="{if (@size) then @size else if
(@maximal-size) then @maximal-size else 'unbounded'}">
        <xsl:apply-templates select="*" mode="nestedTypes"/>
      </xs:sequence>
    </xs:complexType>
  </xsl:template>

  <xsl:template match="*" mode="nestedTypes">
    <xsl:message terminate="yes">
      SFD '<xsl:value-of select="sfd:getOwnNs(.)"/>' ERROR: unknown global type '<xsl:value-of
select="name()"/>'
    </xsl:message>
  </xsl:template>

  <xsl:template match="*" mode="globalTypes">
    <xsl:message terminate="yes">
      SFD '<xsl:value-of select="sfd:getOwnNs(.)"/>' ERROR: unknown global type '<xsl:value-of
select="name()"/>'
    </xsl:message>
  </xsl:template>

  <xsl:template match="xmlFragment">
    <xs:any maxOccurs="1" minOccurs="1" processContents="skip"/>
  </xsl:template>

  <xsl:template match="type-ref" mode="nestedTypes">
    <xs:element name="{if (@name) then @name else sfd:getTypeNameFromSfdRef(@ref)}"
            type="{sfd:removeOwnNs(sfd:getOwnNs(.), sfd:getTypeRefFromSfdRef(sfd:getOwnNs(.), @ref))}"/>

    <!--
             type="{sfd:removeOwnNs(., sfd:getTypeRefFromSfdRef(sfd:getOwnNs(.), @ref))}"/>
    -->
  </xsl:template>

  <!-- templates for the port types / operations
=========================================== -->

  <xsl:template match="mf" mode="port-types">
    <xsl:element namespace="{$wsdlNS}" name="portType">
      <xsl:attribute name="name" select="sfd:getPortTypeName(@type, @name)"/>
      <xsl:apply-templates select="*" mode="operations"/>
    </xsl:element>
```

```
  </xsl:template>

  <xsl:template match="property" mode="operations">
    <xsl:apply-templates select="." mode="property-get"/>
    <xsl:apply-templates select=".[@write-access!='none']" mode="property-set"/>
  </xsl:template>

  <xsl:template match="command" mode="operations">
    <!-- is sfd there is a reference from property to type in property/@ref -->
    <xsl:element namespace="{$wsdlNS}" name="operation">
      <xsl:attribute name="name" select="sfd:getCommandOperationName(@name)"/>

      <xsl:variable name="inMsg">
        <xsl:choose>
          <xsl:when test="in">
            <xsl:value-of select="sfd:getMessageName(sfd:getElementRefFromSfdRef(sfd:getOwnNs(.), in/@ref))"/>
          </xsl:when>
          <xsl:otherwise>mgws:EmptyMessage</xsl:otherwise>
        </xsl:choose>
      </xsl:variable>

      <xsl:variable name="outMsg">
        <xsl:choose>
          <xsl:when test="out">
            <xsl:value-of select="sfd:getMessageName(sfd:getElementRefFromSfdRef(sfd:getOwnNs(.), out/@ref))"/
>
          </xsl:when>
          <xsl:otherwise>mgws:EmptyMessage</xsl:otherwise>
        </xsl:choose>
      </xsl:variable>

      <xsl:element namespace="{$wsdlNS}" name="input">
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., $inMsg)"/>
      </xsl:element>
      <xsl:element namespace="{$wsdlNS}" name="output">
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., $outMsg)"/>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <xsl:template match="property" mode="property-get">
    <!-- is sfd there is a reference from property to type in property/@ref -->
    <xsl:element namespace="{$wsdlNS}" name="operation">
      <xsl:attribute name="name" select="sfd:getPropertyGetOperationName(@name)"/>
      <xsl:variable name="msg" select="sfd:getMessageName(sfd:getElementRefFromSfdRef(sfd:getOwnNs(.),
@ref))"/>
      <xsl:element namespace="{$wsdlNS}" name="input">
        <xsl:attribute name="message" select="'mgws:EmptyMessage'"/>
      </xsl:element>
      <xsl:element namespace="{$wsdlNS}" name="output">
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., $msg)"/>
      </xsl:element>
      <xsl:element namespace="{$wsdlNS}" name="fault">              <!-- fault element in WSDL:operation -->
        <xsl:attribute name="name" select="'Fault'"/>
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., 'errorMessage')"/>
      </xsl:element>
      </xsl:element>
  </xsl:template>

  <xsl:template match="property" mode="property-set">
    <!-- is sfd there is a reference from property to type in property/@ref -->
    <xsl:element namespace="{$wsdlNS}" name="operation">
      <xsl:attribute name="name" select="sfd:getPropertySetOperationName(@name)"/>
      <xsl:variable name="msg" select="sfd:getMessageName(sfd:getElementRefFromSfdRef(sfd:getOwnNs(.),
@ref))"/>
      <xsl:element namespace="{$wsdlNS}" name="input">
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., $msg)"/>
```

```
      </xsl:element>
      <xsl:element namespace="{$wsdlNS}" name="output">
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., $msg)"/>
      </xsl:element>
      <xsl:element namespace="{$wsdlNS}" name="fault">              <!-- fault element in WSDL:operation -->
        <xsl:attribute name="name" select="'Fault'"/>
        <xsl:attribute name="message" select="sfd:addOwnNsIfMissing(., 'errorMessage')"/>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <!-- this template is used to ignore the groups -->
  <xsl:template match="*" mode="operations">
    <xsl:apply-templates select="*" mode="#current"/>
  </xsl:template>

  <!-- templates for the generation of messages
================================================ -->

  <xsl:template match="*" mode="messages">
    <!-- when this template is applied then the context is a global type -->
    <xsl:element namespace="{$wsdlNS}" name="message">
      <xsl:attribute name="name" select="sfd:getMessageName(@name)"/>
      <xsl:element namespace="{$wsdlNS}" name="part">
        <xsl:attribute name="name" select="'Body'"/>                    <!-- set WSDL:part name="body" -->
        <xsl:attribute name="element" select="sfd:ref(../../header/@name, @name)"/>
      </xsl:element>
    </xsl:element>
  </xsl:template>

  <!-- templates for the binding
================================================================ -->

  <xsl:template match="*"  mode="binding">
    <xsl:param name="NS"/>
    <xsl:element namespace="{$wsdlNS}" name="binding">
      <xsl:attribute name="name" select="concat(substring-before(@name, '__'), '__Interface')"/>
      <xsl:attribute name="type" select="sfd:ref($NS, @name)"/>
      <soapbind:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <xsl:apply-templates select="*" mode="binding-operation">
         <xsl:with-param name="NS" select="$NS"/>
      </xsl:apply-templates>
    </xsl:element>
  </xsl:template>

  <xsl:template match="*" mode="binding-operation">
    <xsl:param name="NS"/>
    <xsl:element namespace="{$wsdlNS}" name="operation">
      <xsl:attribute name="name" select="@name"/>
      <xsl:variable name="opType">                    <!-- Decision WS-Transfer soapAction get or Put -->
        <xsl:choose>
          <xsl:when test="starts-with(@name, 'Set')">
            <xsl:text>Put</xsl:text>
          </xsl:when>
          <xsl:when test="starts-with(@name, 'Get')">
            <xsl:text>Get</xsl:text>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text>Get</xsl:text>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>

      <soapbind:operation soapAction="{concat('http://schemas.xmlsoap.org/ws/2004/09/transfer/', $opType)}"/
>
      <xsl:element namespace="{$wsdlNS}" name="input">                  <!-- SOAP header declaration -->
       <soapbind:header message="{sfd:ref($NS, 'ResourceURIMessage')}" part="Header" use="literal" />
```

```
      <soapbind:header message="{sfd:ref($NS, 'SelectorSetMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ToMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ReplyToMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ActionMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'MessageIDMessage')}" part="Header" use="literal" />
      <soapbind:body use="literal"/>
    </xsl:element>
    <xsl:element namespace="{$wsdlNS}" name="output">
      <soapbind:header message="{sfd:ref($NS, 'ResourceURIMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'SelectorSetMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ToMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ReplyToMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'ActionMessage')}" part="Header" use="literal" />
      <soapbind:header message="{sfd:ref($NS, 'MessageIDMessage')}" part="Header" use="literal" />
      <soapbind:body use="literal"/>
    </xsl:element>
  </xsl:element>
 </xsl:template>

 <soapbind:body use="literal"/>

<!-- utility functions
==================================================================== -->

 <xsl:function name="sfd:ref">
  <xsl:param name="namespace"/>
  <xsl:param name="localName"/>
  <xsl:value-of select="$namespace"/>
  <xsl:text>:</xsl:text>
  <xsl:value-of select="$localName"/>
 </xsl:function>


 <!-- getOwnNs() will get the own namespace (namespace of this file).
     for this this function needs a context within a node inside
     the <sfd> element to find the header and thus the namespace -

     in: context of the caller, so that the function can find
        the namespace of the sfd file

     ex. sfd:getOwnNs(.) -> 'SYSLOG'
 -->
 <xsl:function name="sfd:getOwnNs">
  <xsl:param name="context"/>
  <xsl:value-of select="string($context/ancestor::sfd[1]/header/@name)"/>
 </xsl:function>


 <!-- getNs() will get the own namespace (namespace of this file).
  for this this function needs a context within a node inside
  the <sfd> element to find the header and thus the namespace -

  in: context of the caller, so that the function can find
  the namespace of the sfd file

  ex. sfd:getOwnNs(.) -> 'SYSLOG'
 -->
 <xsl:function name="sfd:getQualifiedName">
  <xsl:param name="context"/>
  <xsl:value-of select="string($context/ancestor::sfd[1]/header/@name)"/>
 </xsl:function>
```

```
<!-- removeOwnNs() checks whether the current ref has the (target)
     namespace of this file. If that is the case, the namespace
     prefix will get removed. Otherwise it remains:

     assume we are in a file with namespace prefix 'SYSLOG'

     SYSLOG:Something  ->  Something
     RADIUS:Something  ->  RADIUS:Something
     Something         ->  not allowed, muts be qualified name

     in: own namespace, i.e. 'SYSLOG'
     in: a qualified name like 'SYSLOG:Facility'
-->

<xsl:function name="sfd:removeOwnNs">
  <xsl:param name="ownNs"/>
  <xsl:param name="qualName"/>
  <xsl:variable name="refNs" select="substring-before($qualName, ':')"/>
  <xsl:value-of select="if ($ownNs = $refNs) then sfd:getLocalName($qualName) else $qualName"/>
</xsl:function>


<!-- addOwnNsIfMissing() checks whether the current ref has a namespace. if
     not a given namespace is added

  in: context of the caller, so that the function can find the namespace of the sfd file
  in: a reference like 'syslog:Destination'
-->
<xsl:function name="sfd:addOwnNsIfMissing">
  <xsl:param name="context"/>
  <xsl:param name="ref"/>
  <xsl:variable name="ownNS" select="sfd:getOwnNs($context)"/>
  <xsl:variable name="lName" select="substring-after($ref, ':')"/>
  <xsl:value-of select="if (string-length($lName) > 0) then $ref else sfd:makeQualifiedName($ownNS, $ref)"/>
</xsl:function>


<!-- getElementRefFromSfdRef() will take a type-ref reference like 'SYSLOG::Facility' and converts
  it into a XSD reference like 'SYSLOG:Facility'. It also works if the namepace
  is missing

  in: namespace of this file
  in: a type reference (from type-ref/@ref)

-->
<xsl:function name="sfd:getElementRefFromSfdRef">
  <xsl:param name="localNs"/>
  <xsl:param name="ref"/>
  <xsl:variable name="prefix" select="sfd:getNsPrefixFromSfdRef($ref)"/>
  <xsl:variable name="name"   select="sfd:getElementNameFromSfdRef($ref)"/>
  <xsl:value-of select="sfd:makeQualifiedName($prefix, $localNs, $name)"/>
</xsl:function>


<!-- getTypeRefFromSfdRef() will take a type-ref reference like 'SYSLOG::Facility' and converts
     it into a XSD reference like 'SYSLOG:Facility__Type'. It also works if the namepace
     is missing:

     in: namespace of this file
     in: a type reference (from type-ref/@ref)

     Assume we are in a file with target namespace prefix = "SYSLOG"

     SYSLOG:Facility -> SYSLOG:Facility__Type
     Facility        -> SYSLOG:Facility__Type
-->
<xsl:function name="sfd:getTypeRefFromSfdRef">
```

```
  <xsl:param name="localNs"/>
  <xsl:param name="ref"/>
  <xsl:variable name="prefix"   select="sfd:getNsPrefixFromSfdRef($ref)"/>
  <xsl:variable name="name"     select="sfd:getTypeNameFromSfdRef($ref)"/>
  <xsl:value-of select="sfd:makeQualifiedName($prefix, $localNs, $name)"/>
 </xsl:function>


 <!-- makeQualifiedName() takes a namespace and a local name and makes
     a qualified name out of it:

     in : a namespace prefix
     in : a local name
     out: qualified name

     makeQualifiedName('SYSLOG', 'Facility') -> 'SYSLOG:Facility'
     makeQualifiedName('',       'Facility') -> 'Facility'

     in fact, this is only handling the colon ':'. Otherwise a simple
     concat would do.
 -->
 <xsl:function name="sfd:makeQualifiedName">
  <xsl:param name="nsPrefix"/>
  <xsl:param name="lName"/>
  <xsl:value-of select="if ( '' = $nsPrefix ) then $lName else concat($nsPrefix, ':', $lName)"/>
 </xsl:function>


 <!-- makeQualifiedName() takes a namespace and a local name and makes
  a qualified name out of it:

  in : a namespace prefix
  in : own (this file's) namespace prefix
  in : a local name
  out: qualified name

  makeQualifiedName('RADIUS', 'SYSLOG', 'Facility') -> 'RADIUS:Facility'
  makeQualifiedName('',       'SYSLOG', 'Facility') -> 'SYSLOG:Facility'

  in fact, this is only handling the colon ':'. Otherwise a simple
  concat would do.
 -->
 <xsl:function name="sfd:makeQualifiedName">
  <xsl:param name="nsPrefix"/>
  <xsl:param name="ownPrefix"/>
  <xsl:param name="lName"/>
  <xsl:value-of select="if ( '' = $nsPrefix ) then concat($ownPrefix, ':', $lName) else concat($nsPrefix, ':',
$lName)"/>
 </xsl:function>


 <!-- getLocalName() takes a qualified or local name and returns the
     local name.

     in : a qualified or local name
     out: local name

     getLocalName('SYSLOG:Facility') -> 'Facility'
     getLocalName('Facility'       ) -> 'Facility'
 -->
 <xsl:function name="sfd:getLocalName">
  <xsl:param name="name"/>
  <xsl:value-of select="if ( contains( $name, ':') ) then substring-after($name, ':') else $name"/>
 </xsl:function>
```

```
<!-- getElementNameFromSfdRef() will take a type-ref reference like 'SYSLOG::Facility' and
  returns the name of the type like 'Facility__Type'

  in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getElementNameFromSfdRef">
  <xsl:param name="ref"/>
  <xsl:variable name="name" select="substring-after($ref, '::')"/>
  <xsl:value-of select="if (string-length($name) = 0) then $ref else $name"/>
</xsl:function>

<!-- getTypeNameFromSfdRef() will take a type-ref reference like 'SYSLOG::Facility' and
     returns the name of the type like 'Facility__Type'

     in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getTypeNameFromSfdRef">
  <xsl:param name="ref"/>
  <xsl:variable name="name" select="substring-after($ref, '::')"/>
  <xsl:value-of select="if ($name = '') then sfd:getTypeName($ref) else sfd:getTypeName($name)"/>
</xsl:function>

<!-- getNsPrefixFromSfdRef() will take a type-ref reference like 'SYSLOG::Facility' and
     returns the prefix like 'SYSLOG'. If there is not namespace like 'SYSLOG::' then
     an empty string will be returned.

     in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getNsPrefixFromSfdRef">
  <xsl:param name="ref"/>
  <xsl:variable name="namespace" select="substring-before($ref, '::')"/>
  <xsl:value-of select="if (string-length($namespace) = 0) then '' else $namespace"/>
</xsl:function>

<!-- getTypeName() will take an sfd type name and convert it into a XSD type
     by adding a suffix like '__Type'

     in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getTypeName">
  <xsl:param name="name"/>
  <xsl:value-of select="concat($name, '__Type')"/>
</xsl:function>

<!-- getMessageName() will take an sfd type name and convert it into a WSDL message
  by adding a suffix like '__Message'

  in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getMessageName">
  <xsl:param name="name"/>
  <xsl:value-of select="concat($name, '__Message')"/>
</xsl:function>

<!-- getPortTypeName() will take an sfd type name and convert it into a wsdl
     portType by adding a suffix like '__PortType

  in: a type reference (from type-ref/@ref)
-->
<xsl:function name="sfd:getPortTypeName">
  <xsl:param name="mfType"/>
  <xsl:param name="mfName"/>
  <xsl:value-of select="concat($mfType, '_', $mfName, '__PortType')"/>
</xsl:function>

<!-- getBindingName() will take an sfd type name and convert it into a wsdl
     binding by adding a suffix like '__Binding'
```

```
  in: a type reference (from type-ref/@ref)
 -->
 <xsl:function name="sfd:getBindingName">
  <xsl:param name="mfType"/>
  <xsl:param name="mfName"/>
  <xsl:value-of select="concat($mfType, '_', $mfName, '__Binding')"/>
 </xsl:function>

 <!-- getPropertyGetOperationName() returns the wsdl operation name for
      the property get operation

  in: the property name
 -->
 <xsl:function name="sfd:getPropertyGetOperationName">
  <xsl:param name="property"/>
  <xsl:value-of select="concat('Get', $property, '__Operation')"/>
 </xsl:function>

 <!-- getPropertySetOperationName() returns the wsdl operation name for
  the property set operation

  in: the property name
 -->
 <xsl:function name="sfd:getPropertySetOperationName">
  <xsl:param name="property"/>
  <xsl:value-of select="concat('Set', $property, '__Operation')"/>
 </xsl:function>

 <!-- getCommandOperationName() returns the wsdl operation name for
  the a command operation

  in: the command name
 -->
 <xsl:function name="sfd:getCommandOperationName">
  <xsl:param name="command"/>
  <xsl:value-of select="concat($command, '__Operation')"/>
 </xsl:function>

</xsl:stylesheet>
```