



SAGA.M31 - Galaxy - User Documentation

Table of Contents:

1 Documentation.....	3
1.1 Introduction.....	3
1.1.1 What is SAGA.M31 - Galaxy?.....	3
1.1.2 How does SAGA.M31 - Galaxy - work?.....	3
1.2 Installation and Startup of Galaxy.....	4
1.2.1 Requirements.....	4
1.2.2 Installation.....	4
1.2.2.1 Contents of the Install package.....	4
1.2.2.2 Setting up the Database.....	4
1.2.2.3 Prepare the Application Server for the connection to MySQL.....	6
1.2.2.4 Deployment.....	6
1.2.2.5 Installation.....	7
1.2.2.6 Next Steps.....	7
1.2.2.7 Support.....	8
1.2.3 Third-Party Software and Licenses.....	8
1.3 Connectors.....	8
1.3.1 Connector Concept.....	8
1.3.1.1 What is a Connector?.....	8
1.3.1.2 Enhancements.....	9
1.3.2 Types.....	10
1.3.2.1 SQL Connector.....	10
1.3.2.2 LDAP Connector.....	11



1.3.2.3 Host Connector.....	14
1.4 Information Container.....	16
1.4.1 What is a Container?.....	16
1.4.2 Container Creation.....	17
1.4.2.1 Step 1 - Basic Data.....	17
1.4.2.2 Step 2 - Defining Output fields.....	17
1.4.2.3 Step 3 - Define needed Input fields.....	18
1.4.2.4 Step 4 - Summary/Saving the Container.....	18
1.4.3 Testing the Container.....	19
1.5 User Management.....	19
1.5.1 Create a new User.....	19
1.5.2 Access Management.....	19
1.6 Interfaces.....	20
1.6.1 The XML-Service.....	20
1.6.1.1 End Point for the XML-Service.....	20
1.6.2 Interface for Web Services.....	20
1.6.2.1 Access to the WSDL-Descriptions of a Container.....	21
1.6.2.2 Usage of the Web Service Interface.....	21



1. Documentation

1.1. Introduction

1.1.1. What is SAGA.M31 - Galaxy?

SAGA.M31 - Galaxy is an "**Information Supplier**" that provides Data from different sources into an **Information Container**. These containers can be retrieved via simple XML Structures (Web Application Service) or as a Web Service (via SOAP).

Because an Information Container can use multiple data sources (like *SQL*, *LDAP*, *3270*) the development cycle for modern application are simplified extremely. There is no need to care on the data layer so the focus is simply on the business logic.

The development cycle is shortened extremely, migration paths are reduced which provides high cost saving potential. To keep it short: SAGA.M31 - Galaxy - provides high cost savings by simplifying data access.

1.1.2. How does SAGA.M31 - Galaxy - work?

SAGA.M31 - Galaxy - separates the service in three areas:

- Data Connector
- Connector Request
- Information Container

Basically, SAGA.M31 - Galaxy - acts as an additional layer between the application program and the underlying data-layer. This structure allows it to separate the business logic completely from the data access logic. This follows the basic idea of **SOA** and allows an organization to follow this architecture. If there are changes to the underlying data layer these will under normal circumstance not have any influence to the information Container, that part or Information that will be presented to the application program.

The **Data Connector** establish the connection to the data layer structure and is described and defined explicitly.

A **Connector Request** defines the logical request against the connector and provides information (fields) that can be provided in a container. Finally the **Information Container** builds the interface to a calling application. A request is send against this container and answered with a response.



One major element of SAGA.M31 - Galaxy - is the consistency of the Information Container. Migration was never easier because data that resides on a Mainframe can be moved partially or in total to another platform without changing the Container. It's only the connector that needs to be changed.

Here lies one of the major advantages from SAGA.M31 - Galaxy -. The actual business logic of a application doesn't change.

1.2. Installation and Startup of Galaxy

1.2.1. Requirements

To install SAGA.M31 - Galaxy - the following requirements are needed:

- A J2EE Servlet Container, e.g. the free implementation **Tomcat 5.0.x** or the commercial product **IBM WebSphere 5.1**
- The free Java 2 SDK version 1.4.x from **SUN**
- A working installation of the database **MySQL**, also available for free
- The **MySQL JDBC Driver**, also available for free
- (Optional) Microsoft Office XP Web Services Toolkit 2.0 for integration into MS Office-Applications, available for free

1.2.2. Installation

This section describes the installation of SAGA.M31 - Galaxy -

1.2.2.1. Contents of the Install package

After unpacking the installation package the following files are located in the target directory:

File	Description
Galaxy.war	The actual Galaxy application to be installed in the servlet container
createTables.sql	SQL-Script to create the database structure required for operation
installation.txt	This file in English language as text document

Table 1: Files

1.2.2.2. Setting up the Database



The current release of Galaxy is running using a **MySQL**-Database, which can be downloaded at no costs.

Problems with MySQL version 4.1

Due to internal changes in the implementation of MySQL, which aren't completed up to now, currently only *versions up to 4.0 are supported*.

The database must be installed and configured according to the official documentation from the *MySQL Web site*.

Hinweis:

For Operators using **Apache Web Server** with *PHP*, there exists a excellent administrative tool called **phpMyAdmin**, available for free

Afterwards the following steps needs to be done:

1. database creation (e.g. with the name 'galaxyDatabase')
2. creation of a user (e.g. 'galaxyUser' with corresponding rights)
3. Allocation of a password for the created galaxyUser

In case database and Galaxy-Server are running on two different machines, the following steps should be observed:

1. Both computers must 'see' each other in the network. This can easily be checked using the 'ping' command
2. The just created galaxyUser must be authorized to access the MySQL server from the Galaxy platform
3. This can be checked e.g. using a MySQL Application (e.g. MySQL Commandline-Client) which must be able to connect to the MySQL Server

If the database and the user are created, the tables needs to be set up. For that all queries in the file `createTables.sql` must be processed. This can be done via the MySQL Commandline-Client or with help of tools like phpMyAdmin. Using the Commandline-Client the following command can be executed on UNIX operating systems:

```
$ mysql -u galaxyUser -pgalaxyUserPassword galaxyDatabase < createTables.sql
```

Hinweis:

Upgrade: If upgrading from version 0.6 to the new version, a script called 'update_06_10.sql' is available for use.



1.2.2.3. Prepare the Application Server for the connection to MySQL

The downloaded MySQL JDBC Driver contains a Java archive:
`mysql-connector-java-3.x.xx-ga-bin.jar`. 'x' stands for the used version.

Tomcat

When using Tomcat Application server the file must be stored at the location `TOMCAT_INSTALL_DIR/common/lib` as well as the service has to be restarted.

If the SQL connector of Galaxy should operate on a different database than MySQL, the JDBC-Driver must be installed the same way.

The JDBC Driver for **Oracle** can be downloaded at
http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html.

WebSphere

When using WebSphere Application Server the documentation must be followed on how to create a JDBC data source.

1.2.2.4. Deployment

This section describes the approach to deploy Galaxy on IBM's WebSphere 5.1 and Tomcat 5.0.x. A requirement is, that the server is already installed and running.

Deployment on WebSphere (tested with version 5.1.0)

1. Login to the admin control panel
2. Select applications -> Enterprise Applications
3. A list with all installed applications shows up
4. Select "Install"
5. Select the local File `Galaxy.war` for installation
6. Give a root-context for the application, e.g. `/Galaxy`
7. During installation a "Security Advice" will be shown, just ignore this one and click on next
8. Four further steps are following, just submit them by clicking on 'Next'
9. Server will show that the configuration changed and that they have to be saved
10. Again go to Applications -> Enterprise Applications and select the new application
11. In the section "Configuration" the mode of the "Class loader" must be changed from "PARENT_FIRST" to "PARENT_LAST" and the setting "Setting for WAR-Classloader" from "Module" to "Application"



12. Select the button "Use" and save the master configuration afterwards
13. Now the application has to be started. This can be done at Applications -> Enterprise Applications. The application must be checked with help of the checkbox and activated afterwards
14. After a while the status icon changes from red to green, the startup process is done
15. The application is now accessible at 'http://[your.host]:[port]/Galaxy/wui'

Deployment on Tomcat

1. Login to the "Tomcat Manager", you need a username and a password
2. If no user has been defined, this can be done with help of the file `tomcat-users.xml` in the directory `conf` of the tomcat installation
3. The user needs the rights of the "manager"-group. Further information can be found in the **Tomcat Documentation**
4. The "Tomcat Manager" provides the possibility to upload a WAR-File for deployment
5. With help of this function the file `Galaxy.war` can be uploaded to the server
6. After successful deployment the application should be available right away. This can be checked by calling the URL: `http://[your.host]:[port]/Galaxy/wui`. The Galaxy Login-/Install page should show up

1.2.2.5. Installation

On the first call of Galaxy using the URL: `http://[your.host]:[port]/Galaxy/wui` the welcome page for the Galaxy installation should show up

1. Step 1: MySQL Information

Informations for the MySQL database like hostname, port, username, password and database

1. Step 2: MySQL Check

Galaxy tries to connect to the MySQL database using the information from step 1

1. Step 3: Checking Tables

In step 3 Galaxy checks if all tables are in the database given in step 1

1. Step 4: Additional Properties

A Host connector can be set up, for this a hostname, port and (optional) a port for the TN3270Proxy is needed

1.2.2.6. Next Steps



after Galaxy is installed successfully, the following actions can be executed:

1. Creating Connectors
2. Configuration of Requests against one of the connectors
3. Creating Containers which provide fields for the requests
4. Testing the Containers using the "Sample Run"-Function
5. Creating Users and allocating rights for the Containers

A WSDL-File is generated for every Container, so every Container is available for a Web Service!

Tutorials for the first Steps with Galaxy can be located at the **Galaxy Web site** (<http://galaxy.sagadc.com>) under *Documents -> Examples*. This section will be always updated, so good to look by once in a while...

1.2.2.7. Support

If there are any problems during the Installation or Usage of Galaxy, please visit our **Galaxy Forum** (<http://www.sagadc.org/system/forum/index.php>). There can be asked any questions about Galaxy, which will be answered conscientious. We also provide a Email-Address: galaxy@sagadc.com.

1.2.3. Third-Party Software and Licenses

SAGA.M31 - Galaxy - is using Software, which are published under the following licenses:

- Apache Software License, Version 2.0
- Apache Software License, Version 1.1
- Dom4J License
- Binary Code License Agreement
- Gnu Lesser General Public License
- ODMG License

The complete license texts are stored in the "licenses" folder of the installation package.

1.3. Connectors

1.3.1. Connector Concept

1.3.1.1. What is a Connector?



The **connector** is one of the basic elements of Galaxy. It represents an Interface to any Datasource.

To be able to access different Datasources Galaxy is using different types of connectors. In **Galaxy Version 1.0** the following connectors are ready for use:

- **SQL** - Access to databases using the *JDBC-Interface*
- **LDAP** - Access to directory services
- **Host/3270** - Access to 3270 terminal applications

Every of those connector types can exist in multiple instances, to access different data sources for a type (e.g. databases). Those instances again have, depending on the configuration, multiple queries against the specific data source. In case of SQL-Connectors e.g. a query equals an SQL-Query which will be put against the defined database.

The result of such a connector query consists of output fields which may depend on certain input fields.

Example: To read a specific data record (multiple output fields), an ID (a input field) is required

The following data types for fields are ready to use:

- **simple fields** - will be returned as an simple string
- **tables** - table arrangement of *simple fields* to display a query with many data records

All provided connectors can be created and managed using the Galaxy wui (**Web User Interface**).

1.3.1.2. Enhancements

Connectors are implemented as an extendable interface inside of Galaxy. Which means with help of a java implementation it is no problem to provide new connector types. This is one of the strong sides of the connector concept and shows the flexibility of the implementation of Galaxy. The fact that the fields of a connector are not given to outside world directly but assigned to container fields it is possible without great complexity to change a specific field in the background of a connector. This functionality is especially helpful in migration of data sources because the applications built on Galaxy don't have to be modified.

In the future further connector implementations are planned to expand the functional range:

- **SAP** - Access to SAP systems
- **CSV** - Access to simple files in CSV-Format, locally as well as via URL
- **Web Services** - Access to other Web Services
- *...and many more*

1.3.2. Types

1.3.2.1. SQL Connector

Create a SQL Connector

To create a SQL Connector, the link "Create Connector" as to be clicked in the menu. On the following page a list of available connector types is shown. Select type SQL. Clicking on "Go!"-Button will show up a list on parameters which are needed to create a SQL connector:

Parameter	Description
Connectorname	The name for the Connector
Driver Class	Driver Class of the JDBC-Connectors that is available for the target database
Database URL	URL which is used to connect to the database. The structure of the URL is drawn between the different databases and s described in the database specific documentation
Username	Username to authenticate against the database server
Password	Password for the given Username to authenticate on the database server

Clicking on "Create Connector" will create the connector and is ready for configuration.

Base Page of the SQL Connector

The Base Page shows a overview over all existing connectors, queries as well as fields that were configured.

Build a Query

To build a query against the SQL Connector click the link "Add Query" on the SQL



Connector Base Page. This will lead to a configuration page for the new query. Three steps are needed to create a query, which are described here:

Step 1 - Formulate a Query

In the first step the query will be formulated and tested. The query must be inserted in the field called "SQL Query". Clicking "Run Query" will execute the query and shows the result in a table.

To define input fields inside a query, variables in the form of `#[Variable name]#` must be included. Because such statements cannot deliver a meaningful result, the "Parse Query"-Button must be pressed to identify all input fields. A text field is provided for every input field where values must be inserted. The defined variables will be replaced with the given values from the text fields before the query will be sent to the server. Again the result will show up in a table.

The Checkbox "Result always tabular" forces to deliver the result as a table even if its just one data record which was located through the query. If the results are delivered as table, it will performed as just one connector field later on. Unlike if there is only one data record delivered, every field of this data record equals one connector output field.

Step 2 - Define In-/Output fields

In this step, a name for every in-/output field can be assigned that is representing a field out of the connector. In case of a table result, there must be a specific name given to the table, because it's just one connector output field.

Step 3 - Saving the Query

The third step will show the configuration again und provides a button to save the query. at this point the query is configured and ready for use.

1.3.2.2. LDAP Connector

Creation of a LDAP Connector

To create a LDAP Connector, click the link "Create Connector" in the menu. Afterwards the available connector types are given in a selection list. Select the connector type LDAP and submit the choice with the button "Go!". A list is shown with parameters that are needed by an LDAP connector:

Parameter	Description
-----------	-------------

Connectorname	A name for the connector
Hostname	Name or IP-Address of the server providing the LDAP-Service. <i>If the connection should run via encrypted SSL the string "ldaps://"</i> must be set in front of the value
Port	Port on which the LDAP-Service is available
Bind DN	The username for authentication against the server. if no value is given anonymous bind to the service is tempted.
Password	The password for the given Bind DN to authenticate

Clicking on "Create Connector" will create the connector and is ready for configuration of requests.

Base Page of the LDAP Connector

On the base page all configurable LDAP-Requests with the assigned connectors are listed. They also provide the option to edit a request.

Create a Request

This section describes the creation of a request

On the base page the link "Create a new Request" must be selected. This will lead to a 3-step configuration page which is described here.

Step 1 - Define Basic Parameters

In step 1 the basic parameters for the requests are configured. Those are:

Parameter	Description
Name	Name of the request
Target Connector	If multiple LDAP-Connectors were created, a connector can be chosen to which the request should run against
Base DN	Base DN for the request. Specifies at which point the searching in the directory should be progressed
Filter	Defines the Filter to use for the searching.

	Variables can be included which are only known during runtime of the request. Such variables must be bordered in two "#"-Signs to be recognized as such variables by Galaxy
Search Scope	<p>Defines the type of the searching to progress. Three types are available for selection:</p> <ul style="list-style-type: none"> • Subtree Scope - All elements below the given Base DN will be searched through • Onlevel Scope - All elements one level below the given Base DN will be searched through • Object Scope - Only searching through the specified <i>Base DN</i> object, no others
Provide as Table	If checked the result will be displayed as a table. This provides the access to multiple elements based on a request.
Error on empty result	This setting is triggered if the request against the LDAP-Server is returning a empty result. If checked a connector error is triggered which is displayed as a error code in the container response.

Step 2 - Define Values for Input fields

In step 2 all variable input fields defined by the filter in step 1 are displayed again with the possibility to insert example values. Does this happen before switching to step 3 the request will be executed with the given example values and the result is displayed as an HTML-Table.

Step 3 - Definition of the Output fields

Outputfields of the LDAP-Connector consists of a name (name of the connector field) as well as a associated attribute. During the request the output fields will be filled with the according attributed ob the returned objects.

A new output field can be created in two different ways:

- **Button "Add Mapping"**

A row with a new assignment will be created. The name of the output field as well as the name of the associated attributed must be inserted.

- **"Add"-Link of the example data**

If all input fields in step 2 were specified, a table with the attributes of the data record returned first is displayed. Every attribute provides a "Add"-Link to add a output field for the attribute. The name of the attribute will be used as a name for the output field automatically but can be edited.

If all necessary attributes are associated the request can be saved using the button "Save Request".

1.3.2.3. Host Connector

Creation of a Host Connector

An instance of a host connector can be created via menu link "Create Connector". In the following screen the connector type must be selected and submitted via the button "Go!". In the next step the following parameters must be inserted:

Parameter	Description
Name	Name for the Connector
Mainframe Server	Hostname or IP-Address of the TN3270-Server
Mainframe Server Port	Port for on which the TN3270-Server is listening

After clicking "Create Connector" the new host connector is created.

Request Creation

Sense of the host connector is to access transactions and applications on 3270 basis with help of the screen scraping technology. For this, the paths have to be defined on which the connector is going to move between the screens of the application. The default screen is the screen shown first after connecting to the application. Based on the previously defined paths the connector is now moving into the application, reading the desired fields and jumps back to the default screen waiting for further requests.

Paths for the connector are created in 2 steps:

- Recording the path with help of a 3270-Emulation
- Editing the recorded paths and defining of in- and output fields

Those steps are elucidated in the following section.

Step 1 - Recording the Path



At first the base page of the 3270-Connector must be invoked. This happens using the link "Host Connector" in the menu.

The base page puts up a controller for the so called 3270-Proxy which operates between a 3270-Emulation and a 3270-Server to record the actions taken by the user as well as the paths passing through.

At first a target connector must be selected for the path. This happens by the given selection list. After that a port must be set on which the proxy is going to listen. Default port given is 2023. If no other application is using this port on the machine where Galaxy is running on the setting doesn't have to be modified.

Clicking on the button "Start Server" will start the server which is now waiting for an incoming 3270-Emulation

Now a 3270-Emulation has to be started and connecting against the configured port on the Galaxy-Server. The proxy is now building a connection to the configured TN3270-Server and passing all data which are coming from the emulation to the server and vice versa.

Clicking on the "Refresh"-Link is going to show an active connection. Simultaneously a text field is provided in which the name of the recorded action must be inserted. A click on "Record" will start the recording process.

Now the path which the connector should run through automatically later on must be followed with help of the emulation. At the end of the recording the screen which was shown at the start of the emulation has to be visible.

Clicking on "Stop Recording" will stop the recording process and saving all steps in a database.

Hinweis:

The protocol TN3270E is currently not supported, this option must be deactivated in the emulation.

Step 2 - Defining In-/Output fields

A click on the link "action list" is leading to a list of all recorded paths up to now. There a path has to be selected for which in- and output fields are going to be assigned.

A Java-Applet will open showing the previously recorded screens for post-processing.

For every screen the following actions have to be done:

- mark "variable areas", these are areas which can change by running through again (e.g. clock times, LO_Names or fields from shown data records). This is used to recognize the screens by running through the path. To select the elected area as "variable", right-click on the mouse, a menu should pop up, there choose "variable".
- Defining of output fields, these are areas that should be available as connector fields later on. To select a selected area as an output field, right-click with the mouse and choose "Output". In the table on the right side of the applet, enter a name in the column "Name".
- Using the same principle to define input fields, these are areas used to insert data from a connector input field (e.g. selection of a data record based on a ID-Number).

Is the path on a screen, where no information are needed, the system must be informed of that with help of the button "Switch direction". From this point on the terminal is taken to its initial state (first screen) while the requested data is on its way for delivery to the container.

Are these steps completed, the path has to be saved using the "Save Action"-Button. Now its ready for use. The defined fields are now available as connector in- and output fields and can be packaged into a container.

1.4. Information Container

1.4.1. What is a Container?

Information Container are placing the core concept of SAGA.M31 - Galaxy -. They provide a unique, consistent and defined interface for integration of the data in any application.

Containers are defined by their In- and Output fields, presenting the parameters for a query as well as the return value. Containers itself only define the interface not the actual data source.

To fill a container with data, it fields are associated with any connectors. Using this abstraction a container provides flexible data access without the necessity to know the actual data source. During a migration e.g. the fields of a container just have to be associated with other connectors. Because the interface doesn't change on the outside, no applications built on Galaxy have to be modified.



SAGA.M31 - Galaxy - Information Container can be created, configured and maintained using the Galaxy wui (**Web User Interface**)

Containers are describes with help of an **WSDL** definition and provided as an **Web Service** which enables easy integration in any programming language and applications.

During the implementation of the Web Service Interfaces compatibility was a major element like for example the **Document/Literal - Style** was chosen to support integration into different frameworks (e.g. **Microsoft's .NET**).

1.4.2. Container Creation

To create a container the link "Create Container" has to be chosen from the menu. The container will be created with help of a 4-way action which is described below.

1.4.2.1. Step 1 - Basic Data

Table of basic data for a container:

parameter	description
Name	Name of the container
Handle	A speaking name. This will used while calling the different Service-Interfaces of Galaxy to specify the target container for this query.
Description	Input field where the function of this container can be described.
WSDL Public available	If checked, WSDL which describes this container is available via HTTP request. This has the advantage that you can integrate the most toolkits for web services without detours. Through this the interface of the connector is known to the public which maybe isn't desired in terms of security. If not checked, the WSDL must be downloaded, saved locally and integrated using the user interface of Galaxy

1.4.2.2. Step 2 - Defining Output fields

In this step all fields the container want to provide to outside have to be defined For this a list with all available fields from the different connectors are shown, which can

be added as an container field using the "Add"-Link.

To add a field, first you have to select a connector from the selective list "Fields in Connector" which provides the desired field. After selecting a connector all fields this connector provides are shown in a table.

With help of the left shown list of the defined query in the connector you can filter the list to make it easy to search for a specific field.

Is the connector field found, click the "Add"-Link to add this field to the container fields (lower table). For every field added this way, a name can be defined under which it will be delivered.

A "Re map" function is available for each field in the container field table. Using this link the field will be mapped new to another connector field without the necessity to modify the interface of the container itself. One on this way selected container field will be marked coloured and the "Add"-Link of the connector fields replaces the marked field instead of adding it new. Another click on "Re map"-Link will cancel the coloured mark.

A click on the button "Next Step" implements a allocation of the needed input fields.

1.4.2.3. Step 3 - Define needed Input fields

In this section all connector fields are shown which depends on the selected output fields. Those fields also have to be added to the container fields. For this use the link "Add as new" which will define a new container field as described above, but this time it's a input field.

Every connector field, which depends on one of the given output fields has to be assigned that way

Once defined input fields can be remapped using the "Re map"-Link, like the output fields.

1.4.2.4. Step 4 - Summary/Saving the Container

This section again shows the created configuration as a summary. If all details are correct the container can be saved with a click on "Save Container". From now on this container is ready for use.

A new container should be tested with help of the function "Sample Run" to be sure that the container provides the correct data. This will described in the next section.

1.4.3. Testing the Container

In the container view (reachable via menu link "Manage Containers") a "Sample Run"-Link is available for every container. A click on it leads to the test page.

At first all input fields are shown which are necessary for the container run. For every field a test field is provided in which a value for the test run must be entered.

A click on "Fire!"-Button will start the query and the returned fields are displayed in a HTML-Table.

To view the structure for XML-Request and XML-Response for this container the checkbox "Show XML Request/Response" must be checked. After execution the structures below the returned container fields are displayed.

To analyse errors during execution there is a checkbox called "Show the Trace". If checked detailed information about the processing inside the connectors are shown. In case of a SQL connector e.g. which SQL statements were sent concrete as well as the returned data records. The format of this output varies among the different connector types.

1.5. User Management

1.5.1. Create a new User

The link "Create User" in the menu leads to a form with whose help a new user can be created. The following fields must be filled:

- Username
- Password (incl. Confirmation)
- First Name
- Last Name
- Email Address
- Role of the User

Clicking on "Save" will create the new user. Now the containers needs to be defined on which the new user should have access to.

1.5.2. Access Management

A click on the link "Permissions" in the user overview (link "Edit/Delete User" in the menu) leads to the authorization management. Every single container can be



activated to be usable by a user. Clicking on "Save Changes"-Button will save the settings.

1.6. Interfaces

1.6.1. The XML-Service

The XML-Service from SAGA.M31 - Galaxy - presents a simple access mechanism to one of the defined information container.

With help of a **HTTP-Post-Request** a request in *XML* mode is sent to the Galaxy server which responses with the supplied data fields, again in XML-Structure. The requesting application can post-process those data in any way.

The structure (request and response) for a certain container can be displayed with help of a **Sample Run** and adopted from an application. For this the option "*Show XML Request/Response*" must be checked as well as the *Version 1 (V1)* of the XML-Structure. After clicking the "Fire!"-Button the connector will be executed and the structures below the output fields are displayed.

Every request contains a **username** and a **password** to authenticate against the Galaxy server whereas the password must be delivered as MD5-Checksum and not as plaintext.

The response always contains a **return code** with whose help errors can be checked during execution. If the container contains a return code **greater than 0**, the container didn't run through at all or not quite complete. In this case caution is needed while post-processing the returned data. In this case a **error message** (*ContainerRunMessage*) is provided which points to the errors occurred.

1.6.1.1. End Point for the XML-Service

The **HTTP-Post-Request** mentioned above must be fired to a **special URL** which looks like the following:

```
http[s]://[Hostname]:[Port]/Galaxy/xmlService
```

The concrete URL of the *End Point* can be viewed in the user interface by clicking on "**XML via HTTP**".

1.6.2. Interface for Web Services



Additionally to the simple type of the *XML-Service* Galaxy provides an Interface to access container with the help of **Web Services** since *version 0.6*.

For this purpose every container is described with help of a **WSDL**-Definition and accessible using a *SOAP request over HTTP*.

Tools to include Web Services exists more most of the programming languages, this acquires the possibility to integrate galaxy container into applications fast and easy.

A major element while implementing this interface was the compatibility with those development tools, e.g. **Document/Literal Style** was chosen to support implementations like **Microsoft Office XP Web Services Toolkit 2.0** without any problems.

1.6.2.1. Access to the WSDL-Descriptions of a Container

Galaxy provides WSDL's on two different ways:

Access via HTTP

vVia the URL

```
http[s]://[Hostname]:[Port]/Galaxy/wsd1/[ContainerHandle].wsdl
```

the description can be downloaded. For that to work a handle in the container must be given and the container definition must be defined as *public available*.

Access by the User interface

Alternative method is to obtain the **WSDL** link from the container overview (link "*Manage Containers*" in the menu).

This option is available in case it is not desired to reveal the WSDL for unauthorized user.

1.6.2.2. Usage of the Web Service Interface

The WSDL described SOAP-Request must be send to the following URL:

```
http[s]://[Hostname]:[Port]/Galaxy/services/SOAPService
```

The concrete URL is also mentioned in the WSDL und should be taken over by the tools.

The response structure is also described in the WSDL and fit the output fields in the



defined container.