



SAGA.M31 - Galaxy - Benutzerdokumentation

Inhalt:

1 Dokumentation.....	3
1.1 Einleitung.....	3
1.1.1 Was ist SAGA.M31 - Galaxy?.....	3
1.1.2 Wie arbeitet SAGA.M31 - Galaxy?.....	3
1.2 Installation und Inbetriebnahme von Galaxy.....	4
1.2.1 Voraussetzungen.....	4
1.2.2 Installation	4
1.2.2.1 Inhalt des Installationspaketes.....	4
1.2.2.2 Einrichten der Datenbank.....	5
1.2.2.3 Bearbeiten der Galaxy Konfigurationsdatei.....	6
1.2.2.4 Vorbereiten des Application Servers für die Verbindung zu MySQL.....	7
1.2.2.5 Deployment.....	7
1.2.2.6 Nächste Schritte.....	9
1.2.2.7 Support.....	9
1.2.3 Fremdsoftware und Lizenzen.....	9
1.3 Connectoren.....	10
1.3.1 Das Connectorenkonzept.....	10
1.3.1.1 Was ist ein Connector?.....	10
1.3.1.2 Erweiterungen.....	11
1.3.2 Typen.....	11
1.3.2.1 SQL Connector.....	11
1.3.2.2 LDAP Connector.....	13

1.3.2.3 Host Connector.....	16
1.4 Informationscontainer.....	18
1.4.1 Was ist ein Container?.....	19
1.4.2 Erstellen eines Containers.....	19
1.4.2.1 Step 1 - Basisdaten.....	19
1.4.2.2 Step 2 - Ausgabefelder definieren.....	20
1.4.2.3 Step 3 - Benötigte Eingabefelder definieren.....	21
1.4.2.4 Step 4 - Zusammenfassung/Speichern des Containers	21
1.4.3 Testen des Containers.....	21
1.5 Benutzerverwaltung.....	22
1.5.1 Anlegen eines Benutzers.....	22
1.5.2 Zugriffsverwaltung.....	22
1.6 Schnittstellen.....	23
1.6.1 Der XML-Service.....	23
1.6.1.1 Endpunkt für den XML-Service.....	23
1.6.2 Die Web Services Schnittstelle.....	24
1.6.2.1 Zugriff auf die WSDL-Beschreibungen eines Containers.....	24
1.6.2.2 Benutzung der Web Services Schnittstelle.....	24



1. Dokumentation

1.1. Einleitung

1.1.1. Was ist SAGA.M31 - Galaxy?

SAGA.M31 - Galaxy ist ein "**Datenlieferant**" der Informationen verschiedener Herkünfte in **Informationscontainern** zusammenfasst. Diese Container können über XML-Strukturen (*Web Application Service*) oder als Web Service angefordert werden.

Da in einem Informationscontainer mehrere Datenherkünfte (z.B. *SQL, LDAP, 3270*) zusammengefasst werden können, vereinfacht sich die Entwicklung moderner Applikationen erheblich, da keine Rücksicht mehr auf die dem Anwendungsprogramm zu Grunde liegende Datenarchitektur genommen werden muss.

Die Entwicklungszeiten verringern sich enorm, die Migrationspfade verkürzen sich wodurch sich insgesamt ein enormes Einsparpotential realisieren lässt. Kurz gesagt: SAGA.M31 - Galaxy - ermöglicht erhebliche Kosteneinsparung durch vereinfachte Datenzugriffe.

1.1.2. Wie arbeitet SAGA.M31 - Galaxy?

SAGA.M31 - Galaxy - unterteilt den Service in 3 Bereiche:

- Connectoren
- Connectorenabfragen
- Informationscontainer

Im Grunde genommen legt sich SAGA.M31 - Galaxy - als eine logische Schicht zwischen das Anwendungsprogramm und der darunter liegenden Datenherkunftsschicht. Durch diese Struktur ist es möglich, die Anwendungslogik von der Datenzugriffslogik vollständig zu entkoppeln und somit dem Grundgedanken von **SOA** tatsächlich zu folgen. Sollten sich Änderungen an der Datenherkunftsschicht ergeben, so haben diese in der Regel keinen Einfluss auf den Information Container, also den Teil der Information, den eine Anwendung präsentiert bekommt.

Der **Connector** bildet die Verbindung zur Datenherkunftsschicht und wird explizit beschrieben und definiert.



Eine **Connectorenabfrage** stellt die logische Abfrage gegen einen Connector dar und bietet Informationen (Felder) zur Bereitstellung in einem Container an. Letztlich der **Informationscontainer**. Dieser ist die Schnittstelle zum anfordernden Programm. Gegen diesen Container wird ein Request gesendet, der mit einer Response beantwortet wird.

Wir haben bei der Entwicklung Wert darauf gelegt, dass der Informationscontainer eine konsistente Antwort darstellt. Somit ist es möglich, von einem Mainframe bereitgestellte Informationen nach einer Migration von Teilen der Daten, diese zum Beispiel aus einer SQL Umgebung bereit zu stellen, ohne dass sich die Schnittstelle des Informationscontainers ändert.

Hier liegt eins der wesentlichen Vorteile von SAGA.M31 - Galaxy -. Die eigentliche Geschäftslogik des Anwendungsprogramm ändert sich nicht.

1.2. Installation und Inbetriebnahme von Galaxy

1.2.1. Voraussetzungen

Folgendes wird benötigt, um SAGA.M31 - Galaxy zu installieren:

- Ein J2EE Servletcontainer, z.B. die kostenlose Implementierung **Tomcat 5.0.x** oder das kommerzielle Produkt **IBM WebSphere 5.1**
- Das kostenlose Java 2 SDK in der Version 1.4.x von **SUN**
- Eine Installation der Datenbank **MySQL**, ebenfalls kostenlos
- Den **MySQL JDBC Treiber**, frei verfügbar.
- Optional, zur Integration von Daten aus Galaxy in MS Office-Anwendungen **Microsoft Office XP Web Services Toolkit 2.0**(kostenlos)

1.2.2. Installation

Dieser Abschnitt beschreibt die Installation von SAGA.M31 - Galaxy

1.2.2.1. Inhalt des Installationspaketes

Nach entpacken des Installationspaketes befinden sich die folgenden Dateien im Zielverzeichnis:

Datei	Beschreibung
Galaxy.war	Die eigentliche Galaxy Applikation die im Servletcontainer eingespielt wird
createTables.sql	SQL-Skript zum erstellen der für den Betrieb

	notwendigen Tabellenstruktur
installation.txt	Diese Datei in englischer Sprache als Textdokument
WEB-INF\classes\galaxy.properties	Die Vorlage einer Galaxy-Konfigurationsdatei

Table 1: Dateien

1.2.2.2. Einrichten der Datenbank

Die aktuelle Version von Galaxy wird mit einer **MySQL**-Datenbank betrieben, die kostenlos herunter geladen werden kann.

Probleme mit MySQL Version 4.1

Durch interne Änderungen in der Implementierung von MySQL, die bis jetzt noch nicht abgeschlossen sind, werden derzeit *nur Versionen bis zur 4.0 Serie unterstützt*.

Die Datenbank muss gemäß der Dokumentation auf der *MySQL Website* installiert und konfiguriert werden.

Hinweis:

Für Betreiber des **Apache Web Server** mit *PHP*, steht ein hervorragendes Administrationstool, **phpMyAdmin**, zur Verfügung, Natürlich kostenlos

Anschließend müssen die folgenden Schritte durchgeführt werden:

1. Erstellen einer Datenbank, z.B. mit dem Namen 'galaxyDatabase'
2. Erstellen eines Benutzers, z.B. 'galaxyUser' mit entsprechenden Rechten
3. Vergabe eines Passwortes für den erstellten galaxyUser

Sollen Datenbank und Galaxy-Server auf getrennten Maschinen laufen, sollten zusätzlich folgende Punkte beachtet werden:

1. Die beiden Rechner müssen sich im Netzwerk 'sehen'. Dies kann einfach mit Hilfe des 'ping' Befehles überprüft werden
2. Der erstellte galaxyUser muss berechtigt sein, von der Galaxy-Plattform auf den MySQL Server zuzugreifen
3. Dies kann z.B. durch eine MySQL Applikation (z.B. dem MySQL Commandline-Client) überprüft werden, der von der Galaxy-Plattform eine Verbindung zum MySQL Server aufbauen können muss.

Sind Datenbank und Benutzer erstellt müssen die benötigten Tabellen eingerichtet werden. Hierzu müssen alle Befehle, die in der beiliegenden Datei `createTables.sql` aufgeführt sind abgearbeitet werden. Dies kann entweder per



MySQL Commandline Client erfolgen oder aber mit Hilfe eines Tools wie phpMyAdmin. Bei Benutzung des Commandline Clients unter einem UNIX Betriebssystem kann der folgende Befehl abgesetzt werden:

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

Anschließend kann mit dem Befehl 'show tables' überprüft werden, ob die Tabellen angelegt wurden. Auch kann geprüft werden dass die Tabellen `Users` und `hibernate_unique_key` jeweils einen Eintrag enthalten.

Hinweis:

Upgrade: Wird von der Galaxy Version 0.6 auf die neue Version umgestellt, steht hierfür das Skript 'upgrade_V0_5_V1_1.sql' bereit.

1.2.2.3. Bearbeiten der Galaxy Konfigurationsdatei

Es muss die Datei `galaxy.properties` im Verzeichnis `WEB-INF\classes` editiert werden. Hier müssen die folgenden Werte angepasst werden:

Parameter	Beschreibung
<code>database.url</code>	Die URL unter der die <code>galaxyDatabase</code> erreichbar ist. Die Syntax lautet wie folgt: <code>jdbc:mysql://[hostname ipaddress]:[port]/[databasename]?relaxAutoCommit=true (Ohne Zeilenumbrüche)</code> wobei IP Adresse oder Hostname des Datenbankservers, dessen Port und der Name der angelegten Datenbank (z.B. <code>galaxyDatabase</code>) ersetzt werden müssen.
<code>database.username</code>	Der Name des vorher angelegten Benutzers (z.B. <code>galaxyUser</code>)
<code>database.password</code>	Das Passwort, das für den Benutzer vergeben wurde
<code>hostconnector.terminalType</code>	Der Typ des Terminals, der bei TN-Server angefordert wird. Notwendig, um z.B. eine spezifische LU anzufordern. Dieser Parameter ist nur Relevant, wenn der Hostconnector zum Einsatz kommt.



Nach Anpassung der Parameter muss die Konfigurationsdatei zur Datei `Galaxy.war` hinzugefügt werden.

Hierzu wird die folgende Vorgehensweise empfohlen:

1. Wechsel in das Verzeichnis, in das das Installationspaket entpackt wurde
2. Absetzen des folgenden Befehles:

```
$ jar -uvf Galaxy.war WEB-INF/classes/galaxy.properties
```

Ist dies geschehen, ist die lokale Konfiguration abgeschlossen. Nun müssen dem Application Server die MySQL-JDBC-Treiberklassen bekannt gemacht werden.

1.2.2.4. Vorbereiten des Application Servers für die Verbindung zu MySQL

Der heruntergeladene MySQL JDBC Treiber beinhaltet ein Java Archiv: `mysql-connector-java-3.x.xx-ga-bin.jar`. Hierbei steht 'x' für die verwendete Version.

Tomcat

Bei Verwendung des Tomcat Application Servers muss diese Datei unter `TOMCAT_INSTALL_DIR/common/lib` abgelegt, sowie der Dienst neu gestartet werden.

Soll der SQL Connector von Galaxy auf einer anderen Datenbank als MySQL operieren, so müssen dessen JDBC-Treiber auf die gleiche Art eingespielt werden.

Die JDBC Treiber für **Oracle** können unter http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html heruntergeladen werden.

WebSphere

Bei einem WebSphere Application Server muss den Anweisungen zum erstellen einer JDBC Datenquelle in der Dokumentation gefolgt werden.

1.2.2.5. Deployment

Dieses Dokument beschreibt die Vorgehensweise beim Deployment auf IBM's WebSphere 5.1 und Tomcat 5.0.x. Es wird vorausgesetzt, dass der entsprechende Server bereits installiert und konfiguriert ist.

Deployment auf WebSphere (getestet mit Version 5.1.0)

1. Login in die Administrationskonsole
2. Auswahl des Menüpunktes Anwendungen -> Enterprise Anwendungen
3. Es erscheint eine liste aller installierten Anwendungen
4. Auswahl "Installieren"
5. Auswal der lokalen Datei `Galaxy.war` zur Installation
6. Angeben des Root-Kontextes für die Applikation, z.B. `Galaxy`
7. Während der Installation wird ein "Security Advise" gezeigt, dieser kann durch einen Klick auf weiter ignoriert werden.
8. Vier weitere Schritte folgen, die jeweils nur mit 'Weiter' bestätigt werden müssen
9. Der Server zeigt an, dass sich die Konfiguration geändert hat und diese gespeichert werden muss.
10. Erneut Anwendungen -> Enterprise Anwendungen auswählen und die neu Installierte Anwendung auswählen.
11. Im Abschnitt "Konfiguration" muss der "Modus des Klassenladers" von "PARENT_FIRST" auf "PARENT_LAST" und die "Einstellungen für WAR-Klassenlader" von "Module" to "Application" gestellt werden.
12. Auswahl des Buttons "Anwenden und anschließendes speichern der Masterkonfiguration
13. Nun muss die Anwendung gestartet werden dies kann wieder unter dem Punkt Anwendungen -> Enterprise Anwendungen geschehen. Die Applikation muss mit Hilfe der Checkbox ausgewählt und anschließend gestartet werden.
14. Nach einer Weile schaltet das Statusicon von Rot auf Grün um, der Startvorgang ist abgeschlossen
15. Die Applikation ist nun unter `http://[your.host]:[port]/Galaxy/wui` erreichbar
16. Um dies zu Überprüfen, kann die URL in einen Browser eingegeben und geladen werden, die Loginseite baut sich auf
17. Anmelden mit Benutzer 'admin', Passwort 'galaxy'
18. Es baut sich die Einstiegseite auf, diese enthält erste Informationen zur Benutzung von Galaxy

Deployment on Tomcat

1. Login in den "Tomcat Manager", hierfür wird werden Benutzername und Passwort benötigt
2. Falls noch kein Benutzer definiert wurde, kann dies mit Hilfe der Datei `tomcat-users.xml` im `conf`-Verzeichnis der Tomcatinstallation geschehen
3. Der Benutzer benötigt die Rechte der "manager"-Gruppe. Weitere Informationen finden sich in der **Tomcat Dokumentation**
4. Der "Tomcat Manager" bietet die Möglichkeit, eine WAR-Datei zum deployment hochzuladen
5. Mit Hilfe dieser Funktion muss nun die Datei `Galaxy.war` auf den Server



geladen werden.

6. Nach erfolgreichem Deployment sollte die Applikation bereits verfügbar sein. Dies kann per Aufruf der folgenden URL überprüft werden:
`http://[your.host]:[port]/Galaxy/wui`. Es sollte sich die Galaxy Loginseite zeigen
7. Anmelden mit Benutzername 'admin' und Passwort 'galaxy'
8. Es baut sich die Einstiegseite auf, diese enthält erste Informationen zur Benutzung von Galaxy

1.2.2.6. Nächste Schritte

Nachdem Galaxy nun Installiert ist, können die folgenden Aktionen ausgeführt werden:

1. Erstellen von Connectoren
2. Konfigurieren von Abfragen gegen einen der Connectorer
3. Erstellen von Containern, die Felder der Abfragen bereitstellen
4. Testen der Container über die "Sample Run"-Funktion
5. Erstellen von Benutzern und Vergabe von Rechten für Container

Für Jeden verfügbaren Container wird automatisch eine WSDL-Datei generiert, jeder Container stellt also automatisch auch einen Web Service zur Verfügung!

Tutorials zu den ersten Schritten mit Galaxy finden sich auf der **Galaxy Website** (<http://galaxy.sagadc.com>) unter *Dokumente* -> *Beispiele*. Diese Sektion wird ständig aktualisiert, es lohnt sich also, regelmäßig vorbeizuschauen...

1.2.2.7. Support

Sollten bei der Installation oder der Nutzung von Galaxy Probleme auftreten, empfehlen wir den Besuch der **Galaxy Foren**

(<http://www.sagadc.org/system/forum/index.php>). Dort können Fragen rund um Galaxy gestellt werden, die gewissenhaft beantwortet werden. Wir sind auch per Email unter der folgenden Adresse gerne für Sie da:
galaxy@sagadc.com.

1.2.3. Fremdsoftware und Lizenzen

SAGA.M31 - Galaxy - verwendet Software, die unter den folgenden Lizenzen veröffentlicht wurde:

- 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

Die Vollständigen Lizenztexte sind im "licenses" Ordner des Installationspaketes aufgeführt.

1.3. Connectoren

1.3.1. Das Connectorenkonzept

1.3.1.1. Was ist ein Connector?

Der **Connector** ist einer der Basisbestandteile des Galaxykonzeptes. Er repräsentiert die Schnittstelle zu einer beliebigen Datenquelle.

Um auf verschiedene Datenquellen zugreifen zu können nutzt Galaxy unterschiedliche Typen von Connectoren. In **Galaxy Version 1.0** stehen die folgenden Connectoren bereit:

- **SQL** - Für den Zugriff auf Datenbanken über die *JDBC-Schnittstelle*
- **LDAP** - Für den Zugriff auf Verzeichnisdienste
- **Host/3270** - Für den Zugriff auf 3270 gestützte Terminalapplikationen

Jeder dieser Connector Typen kann in mehreren Instanzen existieren, um auf verschiedene Datenquellen eines Typs (z.B. Datenbanken) zugreifen zu können. Die einzelnen Instanzen wiederum besitzen, je nach Konfiguration, mehrere Abfragen gegen die spezifische Datenquelle. Im Falle des SQL-Connectoren z.B. Entspricht eine Abfrage einer SQL-Query, die gegen die definierte Datenbank gestellt wird.

Das Resultat einer solchen Connectorabfrage besteht aus Ausgabefeldern, die in Abhängigkeit zu bestimmten Eingabefeldern stehen können.

Beispiel: Um einen bestimmten Datensatz (mehrere Ausgabefelder) zu lesen, wird eine ID (ein Eingabefeld) benötigt

Als Datentypen für die Felder stehen derzeit die folgenden Typen bereit:

- **Einfache Felder** - Werden als einfache Zeichenketten zurückgeliefert
- **Tabellen** - Tabellarische Gruppierung von *einfachen Feldern*, zur Abbildung einer Abfrage mit mehreren Datensätzen

Alle mitgelieferten Connectoren können über das Galaxy wui (**Web User Interface**) eingerichtet und konfiguriert werden.

1.3.1.2. Erweiterungen

Connectoren sind innerhalb von Galaxy als erweiterbare Schnittstelle implementiert. Das bedeutet, dass mit Hilfe einer Java Implementierung ohne weiteres neue Connectortypen zur Verfügung gestellt werden können. Dies ist einer der Stärken des Connectorenkonzeptes und zeigt die Flexibilität der Implementierung von Galaxy. Da die Felder eines Connectoren nicht unmittelbar an die Außenwelt weitergereicht werden, sondern Containerfeldern zugeordnet sind, ist es ohne großen Aufwand möglich, den Connector für ein bestimmtes Feld im Hintergrund zu verändern. Diese Funktionalität ist vor allem hilfreich bei Migrationen von Datenquellen, da die Applikationen, die auf Galaxy aufbauen hierbei nicht verändert werden müssen.

Für die Zukunft sind noch weitere Connectorimplementierungen geplant, um den Funktionsumfang von Galaxy zu erweitern:

- **SAP** - Zugriff auf SAP Systeme
- **CSV** - Zugriff auf einfache Dateien im CSV-Format, lokal, sowie per URL
- **Web Services** - Zugriff auf andere Web Services
- *...und viele mehr*

1.3.2. Typen

1.3.2.1. SQL Connector

Erstellen eines SQL Connectors

Um einen SQL Connector zu erstellen, muss im Galaxymenü der Link "Create Connector" betätigt werden. In der folgenden Seite werden die verfügbaren Connectortypen in einer Auswahlliste angeboten, hier muss der Type SQL gewählt werden. Nachdem die Auswahl mit "Go!" bestätigt wurde, wird eine Liste an Parametern angezeigt, die zum erstellen eines SQL Connectoren benötigt werden:

Parameter	Beschreibung
Connectorname	Ein Name unter dem der Connector angesprochen werden kann.
Driver Class	Die Treiberklasse des JDBC-Connectoren der für die Zieldatenbank zur Verfügung gestellt wird.
Database URL	Die URL, die zum Aufbau der Verbindung zum Datenbankserver genutzt werden soll. Der

	Aufbau der URL unterscheidet sich zwischen den verschiedenen Datenbanken und ist in den entsprechenden Dokumentationen beschrieben.
Username	Der Benutzernamen zur Identifikation gegenüber des Datenbankservers.
Password	Das Passwort das den angegebenen Benutzernamen authentifiziert.

Mit einem Klick auf "Create Connector" wird der Connector erstellt und steht zur Konfiguration von Anfragen zur Verfügung.

Die Basisseite des SQL Connectors

Die Basisseite zeigt eine Übersicht aller verfügbaren Connectoren, Abfragen sowie Feldern, die konfiguriert wurden.

Erstellen einer Abfrage

Um eine Anfrage gegen einen SQL Connector zu erstellen muss auf der Basisseite der Link "Add Query" betätigt werden. Dies führt zu einer Konfiguration der Abfrage in drei Schritten, die hier erläutert werden:

Step 1 - Query formulieren

Im ersten Schritt wird die Query formuliert und getestet. In das Feld "SQL Query" ist die Query einzutragen, der Button "Run Query" führt sie aus und zeigt das Ergebnis in einer Tabelle an.

Um nun Inputfelder innerhalb einer Query zu definieren, müssen an den entsprechenden Stellen im Statement Variablen in der Form `#[Variablenname]#` eingefügt werden. Da ein so formuliertes Statement natürlich kein sinnvolles Resultat liefert, müssen zunächst mit Hilfe des "Parse Query"-Buttons alle Inputfelder der Query identifiziert werden. Für jedes Feld wird nun ein Textfeld bereitgestellt, in das ein Wert eingetragen werden muss. Die definierten Variablen werden nun bei der Ausführung durch "Run Query" mit den eingegebenen Ersetzt, bevor die Query an den Server gesendet wird. Wieder wird das Resultat in einer Tabelle angezeigt.

Die Checkbox "Result always tabular" erzwingt, das Ergebnis als Tabelle zurück zu liefern, auch, wenn nur ein einziger Datensatz durch die Query gefunden wird. Werden Resultate als Tabelle zurückgegeben, so wird diese später als ein einziges

Connectorfeld aufgeführt. Wird im Gegensatz hierzu nur ein einziger Datensatz geliefert, so entspricht jedes Feld dieses Datensatzes einem Connectorausgabefeld.

Step 2 - Ein/Ausgabefelder definieren

In diesem Schritt kann für jedes Ein-/Ausgabefeld ein Name vergeben werden, der das Feld außerhalb des Connectoren repräsentiert. Im Falle eines Tabellenresultates muss auch ein Name für die Tabelle spezifiziert werden, da diese als eigenes Connectorausgabefeld dargestellt wird.

Step 3 - Speichern der Abfrage

Der dritte Schritt zeigt noch mal die Konfiguration der Abfrage an und bietet einen Button zum Speichern an. An diesem Punkt ist die Abfrage konfiguriert und Einsatzbereit.

1.3.2.2. LDAP Connector

Erstellen eines LDAP Connectors

Um einen LDAP Connector zu erstellen, muss zunächst der Link der Link "Create Connector" im Galaxy Menü betätigt werden. Anschließend werden die verfügbaren Connectortypen in einer Auswahlliste angeboten, hier muss der Type LDAP ausgewählt und die Auswahl mit "Go!" bestätigt werden. Es wird eine Liste an Parametern angezeigt, die zum erstellen eines LDAP Connectoren benötigt werden:

Parameter	Beschreibung
Connectorname	Ein Name unter dem der Connector angesprochen werden kann.
Hostname	Der Name oder die IP-Adresse des Servers der den LDAP-Dienst bereitstellt. <i>Soll die Verbindung per SSL verschlüsselt werden, so muss dem Wert der String "ldaps://" vorangestellt werden.</i>
Port	Der Port auf dem der LDAP-Dienst zur Verfügung gestellt wird.
Bind DN	Der Benutzername zur Identifikation gegenüber des Servers. Wird dieses Feld leer gelassen, so wird versucht Anonym gegen den Dienst zu binden.

Password	Das Passwort das den angegebenen Bind DN authentifiziert.
----------	---

Mit einem Klick auf "Create Connector" wird der Connector erstellt und steht zur Konfiguration von Anfragen zur Verfügung.

Die Basisseite des LDAP Connectors

Auf der Basisseite werden alle konfigurierten LDAP-Abfragen mit dem zugehörigen Connector aufgelistet. Sie bietet ausserdem die Option, eine der Abfragen zu Editieren.

Erstellen einer Abfrage

Dieser Abschnitt beschreibt das Erstellen einer Abfrage

Auf der Basisseite muss zunächst der Link "Create a new Request" angewählt werden. Dies führt zur Konfiguration der Abfrage in einem drei Schritte Verfahren, das hier erläutert wird.

Step 1 - Basisparameter definieren

In Schritt 1 werden die Basisparameter für die Anfrage konfiguriert. Diese sind:

Parameter	Beschreibung
Name	Name der zu erstellenden Anfrage
Target Connector	Wurden mehrere LDAP-Connectoren erstellt, so kann mit Hilfe dieser Auswahlliste definiert werden, gegen welchen die Abfrage gestellt werden soll.
Base DN	Der Base DN, für die Abfrage. Spezifiziert, ab welchem Knoten die Suche im Verzeichnis durchgeführt werden soll.
Filter	Definiert den anzuwendenden Filter für die Suche. Hier können Variablen eingefügt werden, die erst zur Laufzeit einer Anfrage bekannt sind. Solche Variablen müssen in zwei "#" - Zeichen eingefasst sein um von Galaxy als solche erkannt zu werden.
Search Scope	Definiert die Art der Suche die durchgeführt werden soll. Hier gibt es drei Arten zur Auswahl: <ul style="list-style-type: none"> • Subtree Scope - Alle Elemente unterhalb

	<p>des angegebenen Base DN werden durchsucht.</p> <ul style="list-style-type: none"> • Onlevel Scope - Alle Elemente, die eine Ebene unterhalb des angegebenen Base DN's liegen werden durchsucht. • Object Scope - Durchsucht lediglich das durch <i>Base DN</i> spezifizierte Objekt, keine weiteren.
Provide as Table	Ist diese Checkbox aktiviert, so wird das Ergebnis als Tabelle dargestellt. Dies ermöglicht den Zugriff auf mehrere Elemente basierend auf einer Abfrage.
Error on empty result	Diese Einstellung kommt zum tragen, wenn die Abfrage gegen den LDAP-Server ein leeres Ergebnis zurückliefert. Ist die Checkbox aktiviert, so wird in diesem Fall ein Connectorerror ausgelöst, der in der Containerantwort durch einen Fehlercode abgebildet wird.

Step 2 - Werte für Eingabefelder definieren

In Schritt 2 werden alle variablen Eingabefelder, die im Filter in Schritt 1 definiert wurden nochmals aufgeführt, mit der Möglichkeit der Eingabe von Beispielwerten. Geschieht dies vor dem Wechsel in Schritt 3, so wird die Anfrage mit diesen Werten probeweise ausgeführt, und das Resultat in dort als HTML-Tabelle aufgeführt.

Step 3 - Definition der Ausgabefelder.

Ausgabefelder des LDAP-Connectoren bestehen aus einem Namen (der Name des Connectorfeldes), sowie einem verknüpften Attribut. Bei der Abfrage werden die Ausgabefelder mit den entsprechenden Attributen der zurückgelieferten Objekte gefüllt.

Ein neues Ausgabefeld kann auf zwei verschiedene Arten erstellt werden:

- **Button "Add Mapping"**

Es wird eine Zeile mit einer neue Zuordnung erstellt. Hier müssen Name des Ausgabefeldes sowie der Name des zugeordneten Attributes eingetragen werden.

- **"Add"-Link der Beispieldaten**

Wurden in Schritt 2 Werte für die Eingabefelder spezifiziert, so wird eine Tabelle

mit den Attributen des ersten zurückgelieferten Datensatzes angezeigt. Jedes Attribut verfügt über einen "Add"-Link, über den das Attribut als Ausgabefeld hinzugefügt wird. Als Name für das Feld wird automatisch der Name des Attributes angenommen, dieser kann aber, wenn anders gewünscht von Hand editiert werden.

Sind alle benötigten Attribute verknüpft, kann der Request über den Button "Save Request" gespeichert werden.

1.3.2.3. Host Connector

Erstellen eines Host Connectors

Eine Instanz des Hostconnectors kann über den Menüpunkt "Create Connector" erstellt werden. Im folgenden Schirm muss in der angezeigten Auswahlliste der Connectortyp "Host" ausgewählt und mit dem Button "Go!" bestätigt werden. In nächsten Schritt müssen die Folgenden Parameter eingetragen werden:

Parameter	Beschreibung
Name	Der Name für den neu zu erstellenden Connector
Mainframe Server	Hostname oder IP-Adresse des TN3270-Servers
Mainframe Server Port	Port, auf dem der TN3270-Server lauscht

Ein anschließender Klick auf "Create Connector" erstellt den Connector.

Erstellen einer Abfrage

Sinn des Hostconnectors ist es, mit Hilfe der Screenscraping Technologie auf Transaktionen und Anwendungen auf 3270 Basis zuzugreifen. Zu diesem Zweck müssen die Pfade definiert werden, auf denen sich der Connector zwischen den Schirmen der Anwendung bewegt. Der Ausgangsschirm ist immer der, der nach Aufbau der Verbindung als erstes präsentiert wird. Basierend auf den definierten Pfaden bewegt sich der Connector nun in die Anwendung hinein, ließt dort die gewünschten Felder aus, und springt wieder zurück auf den Ausgangsschirm, um für die nächste Abfrage bereit zu sein.

Pfade für den Connector werden in zwei Schritten erstellt:

- Aufzeichnen des Weges mit Hilfe einer 3270-Emulation
- Bearbeiten des aufgezeichneten Pfades und definition der Ein-und

Ausgabefelder

Diese Schritte werden in diesem Abschnitt erläutert.

Step 1 - Aufzeichnen des Weges

Zunächst muss die Basisseite des 3270-Connectors aufgerufen werden. Dies geschieht über den Link "Host Connector" im Menü.

Die Basisseite stellt die Steuerung für den so genannten 3270-Proxy auf, der sich zwischen eine 3270-Emulation und einen 3270-Server schaltet, um die Aktionen des Benutzers sowie die durchlaufenen Pfade aufzuzeichnen.

Zunächst muss der Zielconnector für den Weg ausgewählt werden. Dies geschieht über die angezeigte Auswahlliste. Anschließend muss ein Port bestimmt werden, auf dem der Proxy lauschen soll. Standardmäßig ist hier der Port 2023 vorgegeben, sollte dieser auf der Maschine, auf der Galaxy läuft noch nicht bereits durch eine andere Anwendung belegt sein, so muss diese Einstellung nicht verändert werden.

Ein Klick auf den Button "Start Server" startet den Server, der nun auf die einkommende Verbindung einer 3270-Emulation wartet.

Nun muss eine 3270-Emulation gestartet und gegen den eingestellten Port auf dem Galaxy-Server verbunden werden. Der Proxy baut nun eine Verbindung zum eingestellten TN3270-Server auf, und leitet alle Daten weiter, die von der Emulation an den Server und umgekehrt gesendet werden.

Ein Klick auf den "Refresh"-Link zeigt an, dass nun eine Verbindung aktiv ist. Gleichzeitig wird ein Textfeld bereitgestellt, in dem der Name der aufzuzeichnenden Aktion eingetragen werden muss. Ein Klick auf "Record" startet nun die Aufzeichnung.

Nun muss mit Hilfe der Emulation genau der Weg beschriftet werden, den später der Connector automatisiert durchlaufen soll. Am Ende der Aufzeichnung muss wieder der Schirm aktiv sein, der nach dem Start der Emulation sichtbar war.

Ein Klick auf den Button "Stop Recording" stoppt die Aufzeichnung und speichert die Schritte in der Datenbank.

Hinweis:

Das Protokoll TN3270E wird derzeit noch nicht unterstützt, in der Emulation muss diese Option also deaktiviert werden.

Step 2 - Ein/Ausgabefelder definieren

Ein Klick auf den Link "action list" führt zu einer Liste aller bisher aufgezeichneten Wege. Dort muss der Weg ausgewählt werden, für den die Ein- und Ausgabefelder bestimmt werden sollen.

Es öffnet sich ein Java-Applet, das die vorher aufgenommenen Schirme zur Nachbearbeitung anzeigt.

In diesem müssen nun für jeden Schirm die folgenden Aktionen durchgeführt werden:

- Markieren der "Variablen Bereiche", dies sind die Bereiche, die sich beim erneuten durchlaufen des Weges ändern können (z.B. Uhrzeiten, LU-Namen, oder Felder von angezeigten Datensätzen). Dies dient zur Wiedererkennung der Schirme beim durchlaufen des Weges. Um den markierten Bereich als "Variabel" zu definieren, muss über die rechte Maustaste der Menüpunkt "Variable" ausgewählt werden.
- Definieren der Ausgabefelder, dies sind Bereiche, die später als Connectorfeld zur Verfügung stehen. Auch dies geschieht mit Hilfe des Kontextmenüs unter dem Punkt "Output". In der Tabelle am rechten Rand des Applets muss in der Spalte "Name" nun noch ein Name eingetragen und mit der Eingabetaste bestätigt werden.
- Nach dem gleichen Prinzip können auch Eingabefelder definiert werden, dies sind die Bereiche, die später aus einem Connectoreingabefeld in die Maske übernommen werden. Ein Beispiel hierfür ist die Selektion eines Datensatzes basierend auf einer Kennnummer.

Ist der Pfad nun an dem Schirm angelangt, ab dem keine Informationen mehr benötigt werden, muss dies dem System mit Hilfe des Buttons "Switch direction" mitgeteilt werden. Ab diesem Punkt wird das Terminal nun asynchron zurück in den Ausgangszustand (erster Schirm) gebracht, während die geforderten Daten bereits zurück an den Container geliefert wurden.

Sind diese Schritte abgeschlossen, so muss der Pfad mit dem "Save Action"-Button gespeichert werden, er steht ab nun zur Benutzung bereit. Die definierten Felder stehen als Connectoren- Ein- und Ausgabefelder bereit und können in einen Container verpackt werden.

1.4. Informationscontainer

1.4.1. Was ist ein Container?

Informationscontainer stellen das Kernkonzept hinter SAGA.M31 - Galaxy - dar. Sie bieten eine einheitliche, konsistente und definierte Schnittstelle für die Integration von Daten in beliebige Anwendungen.

Ein Container wird definiert durch seine Ein- und Ausgabefelder, die die Parameter für eine Abfrage, sowie deren Rückgabewert darstellen. Container selbst definieren hierbei lediglich, die Schnittstelle, nicht die eigentliche Quelle für die Daten.

Um einen Container nun mit Daten zu bestücken, werden seine Felder mit denen eines beliebigen Connectoren verknüpft. Durch diese Abstraktion bietet der Container einen flexiblen Datenzugriff, ohne die Notwendigkeit, die eigentliche Quelle der Daten kennen zu müssen. Bei einer Migration müssen z.B. lediglich die Felder des Containers mit anderen Connectorfeldern verknüpft werden. Da sich die Schnittstelle nach außen hin nicht verändert, müssen die Applikationen, die auf Galaxy aufsetzen, nicht verändert werden.

SAGA.M31 -Galaxy - Informationscontainer können mit Hilfe des Galaxy wui (**Web User Interface**) komfortabel erstellt, konfiguriert und gewartet werden.

Container sind mit Hilfe einer **WSDL** Definition beschrieben und als **Web Service** bereitgestellt, dies ermöglicht die einfache Integratration in beliebige Programmiersprachen und Anwendungen.

Beim der Implementierung der Web Services Schnittstelle wurde besonderen Wert auf Kompatibilität gelegt, so wurde beispielsweise der **Document/Literal - Style** gewählt, um die Integration in verschiedene Frameworks wie z.B. **Microsoft's .NET** zu unterstützen.

1.4.2. Erstellen eines Containers

Um einen Container zu erstellen muss zunächst der Link "Create Container" aus dem Menü ausgewählt werden. Der Container wird mit Hilfe eines 4-Schritte Verfahrens erstellt, das hier beschrieben wird.

1.4.2.1. Step 1 - Basisdaten

Zu den Basisdaten eines Containers zählen:

Parameter	Beschreibung
-----------	--------------

Name	Der Name des Containers
Handle	Ein sprechender Name. Dieser wird beim Aufruf über die verschiedenen Service-Schnittstellen von Galaxy verwendet, um den Zielcontainer für die Abfrage zu definieren.
Description	Ein Textfeld in dem die Funktion des Container beschrieben werden kann.
WSDL Public available	Ist diese Checkbox aktiviert, so ist die WSDL, die diesen Container beschreibt, über eine HTTP Anfrage verfügbar. Dies hat den Vorteil, dass sie ohne Umwege in die meisten Toolkits für Webservices eingebunden werden kann. Hierdurch wird allerdings die Schnittstelle des Containers öffentlich bekannt gegeben, was unter Umständen aus Gründen der Sicherheit nicht erwünscht ist. Ist die Checkbox nicht aktiviert, muss die WSDL über das Benutzerinterface von Galaxy heruntergeladen, lokal gespeichert und dann eingebunden werden.

1.4.2.2. Step 2 - Ausgabefelder definieren

In diesem Schritt müssen die Felder definiert werden, die der Container nach außen hin zur Verfügung stellt. Hierzu wird eine Liste aller verfügbaren Felder der verschiedenen Connectoren angezeigt, die über den "Add"-Link als Containerfeld verfügbar gemacht werden.

Um ein Feld einzufügen muss zunächst in der Auswahlliste "Fields in Connector" der Connector ausgewählt werden, der das gewünschte Feld bereitstellt. Nach der Auswahl werden zunächst alle Felder dieses Connectoren in der Tabelle zur Auswahl angeboten

Mit Hilfe der Links gezeigten Liste der im Connector definierten Abfragen kann die Liste entsprechend der Auswahl gefiltert werden, um die Suche nach einem bestimmten Feld zu erleichtern.

Ist das Connectorenfeld gefunden, wird es über den "Add"-Link zu den Containerfeldern (untere Tabelle) hinzugefügt. Für jedes auf diese Weise hinzugefügte Feld kann dort noch ein Name definiert werden, unter dem das Containerfeld später ausgeliefert wird.

In der Liste der Containerfelder gibt es zu jedem Feld eine "Remap" Funktion, die über den gleichnamigen Link angesprochen wird. Sie dient der Neuordnung eines Feldes zu einem anderen Connectorfeld, ohne hierbei die Schnittstelle des Containers selbst zu verändern. Ein Auf diese Weise ausgewähltes Containerfeld wird farbig markiert, und der "Add"-Link der Connectorfelder ersetzt das markierte Feld anstatt es neu hinzuzufügen. Ein erneuter Klick auf den "Remap"-Link hebt die farbige Markierung wieder auf.

Ein Klick auf den Button "Next Step" führt zur Zuordnung der benötigten Eingabefelder.

1.4.2.3. Step 3 - Benötigte Eingabefelder definieren

In diesem Abschnitt werden zunächst alle Connectorfelder angezeigt, die in Abhängigkeit zu den ausgewählten Ausgabefeldern stehen. Auch diese müssen wiederum Containerfeldern zugeordnet werden. Hierzu steht der Link "Add as new" zur Verfügung, der wie im vorherigen Schritt ein neues Containerfeld definiert, diesmal ein Eingabefeld.

Jedes Connectoreingabefeld, das in Abhängigkeit zu einem der gewählten Ausgabefelder steht, muss auf diese Weise zugeordnet werden.

Einmal definierte Eingabefelder können, wie die Ausgabefelder im letzten Schritt, mit Hilfe der "Remap"-Funktion neuen Connectoreingabefeldern zugeordnet werden.

1.4.2.4. Step 4 - Zusammenfassung/Speichern des Containers

Dieser Abschnitt zeigt nochmals die erstellte Konfiguration als Zusammenfassung an. Sind hier alle Angaben korrekt, so kann der Container mit einem Klick auf "Save Container" gespeichert werden. Von diesem Zeitpunkt an steht er zur Ausführung bereit.

Ein neu erstellter Container sollte zunächst mit Hilfe eines "Sample Run's" getestet werden, um sicher zu gehen, dass der Container die korrekten Daten liefert. Dies wird im nächsten Abschnitt beschrieben.

1.4.3. Testen des Containers

In der Containersicht (erreichbar über den Menülink "Manage Containers") steht für jeden verfügbaren Container ein Link für den "Sample Run" zur Verfügung. Ein Klick darauf führt zur Testseite.

Dort werden zunächst alle Eingabefelder aufgezählt, die für den Containerlauf erforderlich sind. Für jedes Feld wird ein Textfeld bereitgestellt, in das ein Wert für den Probelauf eingegeben werden muss.

Ein Klick auf den "Fire!"-Button startet die Abfrage und die zurückgelieferten Felder werden in einer HTML-Tabelle dargestellt.

Um sich die Strukturen für die XML-Anfrage und die XML-Antwort für diesen Container anzuschauen, muss die Checkbox "Show XML Request/Response" ausgewählt werden. Nach der Ausführung werden die Strukturen unterhalb der zurückgelieferten Containerfelder dargestellt.

Um Fehler, die während der Ausführung auftreten analysieren zu können, gibt es noch die Checkbox "Show the Trace". Ist diese aktiviert, so werden detaillierte Informationen über die Verarbeitung innerhalb der Connectoren angezeigt. Im Falle eines SQL Connectors z.B. welche SQL Statements konkret abgesetzt werden, sowie die zurückgelieferten Daten. Das Format der Ausgabe variiert zwischen den verschiedenen Connectorarten.

1.5. Benutzerverwaltung

1.5.1. Anlegen eines Benutzers

Der Link "Create User" im Menü führt auf ein Formular mit dessen Hilfe ein neuer Benutzer angelegt werden kann. Dort müssen die folgenden Felder ausgefüllt werden:

- Benutzername
- Passwort (plus Bestätigung)
- Vorname
- Nachname
- Email Adresse
- Rolle des Benutzers

Ein Klick auf "Save" legt schließlich den neuen Benutzer an, nun müssen noch die Container definiert werden, auf die der neue Benutzer Zugriff haben soll.

1.5.2. Zugriffsverwaltung

Ein Klick auf den Link "Permissions" in der Benutzerübersicht (Link "Edit/Delete User" im Menü) führt zur Rechteverwaltung. Hier können einzelne Container für Benutzung durch den Benutzer frei geschaltet werden. Ein Klick auf den "Save

Changes"-Button speichert die Einstellungen

1.6. Schnittstellen

1.6.1. Der XML-Service

Der XML-Service von SAGA.M31 - Galaxy - stellt einen einfachen Zugriffsmechanismus auf einen der definierten Informationscontainer dar.

Mit Hilfe eines **HTTP-Post-Requests** wird eine in *XML* formulierte Anfrage an den Galaxyserver abgesetzt, dieser liefert als Antwort die Datenfelder des Containers, ebenfalls als *XML*-Struktur, zurück. Die Aufrufende Applikation kann die Daten nun beliebig weiterverarbeiten.

Die Strukturen (Abfrage und Antwort) für einen bestimmten Container können mit Hilfe eines **Probelaufes** (dem *Sample Run*) angezeigt und entsprechend von einer Anwendung adoptiert werden. Hierbei muss die Checkbox "*Show XML Request/Response*" aktiviert werden, sowie die *Version 1 (V1)* der *XML*-Struktur angewählt werden. Nach einem Klick auf den "Fire!"-Button wird der Container nun ausgeführt und die Strukturen unterhalb des der Ausgabefelder dargestellt.

Jede Anfrage enthält einen **Benutzernamen** und ein **Passwort** um sich gegenüber dem Galaxyserver zu authentifizieren, wobei das Passwort nicht im Klartext, sondern als MD5-Checksumme übergeben werden muss.

In der Antwort ist jeweils ein **Returncode** enthalten, mit dessen Hilfe auf Fehler bei der Ausführung geprüft werden kann. Liefert ein Container einen Returncode **größer als 0**, so ist der Container nicht oder nicht vollständig durchlaufen worden. In diesem Fall ist Vorsicht bei der Weiterverarbeitung der zurückgegebenen Daten geboten. Im diesem Fall steht eine **Fehlermeldung** (die *ContainerRunMessage*) zur Verfügung, aus der auf die Art des aufgetretenen Fehlers geschlossen werden kann.

1.6.1.1. Endpunkt für den XML-Service

Der beschriebene **HTTP-Post-Request** muss hierbei gegen eine **spezielle URL** abgesetzt werden, die sich wie folgt zusammensetzt:

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

Die konkrete URL des *Endpunktes* kann im Userinterface unter dem Menüpunkt **XML via HTTP** abgefragt werden.

1.6.2. Die Web Services Schnittstelle

Zusätzlich zur einfachen Variante des *XML-Services* bietet Galaxy seit der *Version 0.6* auch eine Schnittstelle um mit Hilfe eines **Web Services** auf Container zuzugreifen.

Zu diesem Zweck ist jeder Container mit Hilfe einer **WSDL**-Definition beschrieben und über einen *SOAP Aufruf über HTTP* ansprechbar.

Für die meisten Sprachen existieren mittlerweile Werkzeuge um Web Services einzubinden, dies schafft die Möglichkeit, Galaxy Container leicht und schnell in Applikationen zu integrieren.

Bei der Implementierung dieser Schnittstelle wurde besonders auf die Kompatibilität mit diesen Entwicklungswerkzeugen gelegt, so wurde z.B. der **Document/Literal Style** gewählt, um auch Implementierungen wie **Microsoft Office XP Web Services Toolkit 2.0** ohne Probleme zu unterstützen.

1.6.2.1. Zugriff auf die WSDL-Beschreibungen eines Containers

Galaxy stellt die WSDL's auf zwei verschiedenen Arten zur Verfügung:

Zugriff per HTTP

Über die URL

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

kann die Beschreibung herunter geladen werden. Hierzu muss im entsprechenden Container ein Handle angegeben und die Containerdefinition als *public available* definiert worden sein.

Zugriff über das Userinterface

Alternativ kann die den Link **WSDL** in der Containerübersicht (Link *Manage Containers* im Menü) bezogen werden.

Diese Option steht bereit, falls ein offlegen der WSDL für nicht autorisierte Benutzer nicht gewünscht ist

1.6.2.2. Benutzung der Web Services Schnittstelle

Die In der WSDL beschriebene SOAP-Anfrage muss an die folgende URL gesendet werden:



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

Die konkrete URL ist auch in der WSDL angeführt und wird in der Regel von den Werkzeugen automatisch übernommen.

Die Antwortstrukturen sind ebenfalls in der WSDL beschrieben und entsprechen den im Container definierten Ausgabefeldern.