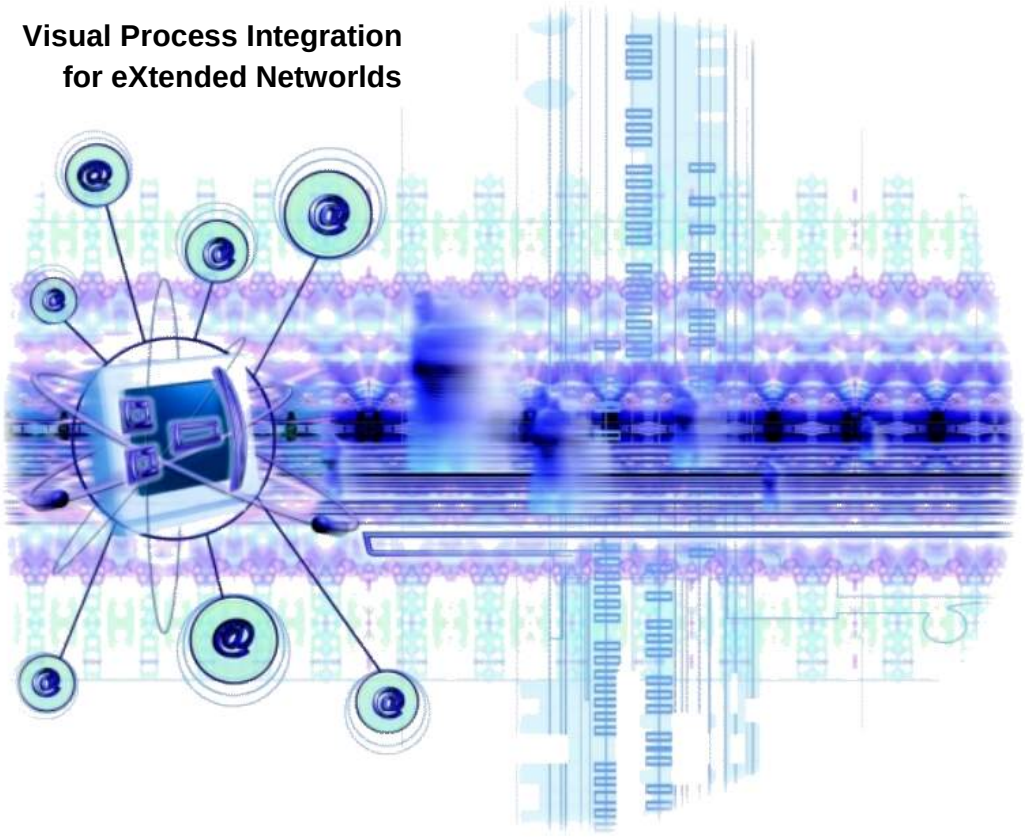
MISSISSIPPI Management

NetXtender Service Description

Visual Process Integration
for eXtended Networlds



Contents

Figures

Documents

Ref.	Datum	Titel / Version	Verantwortlich
[1]		Xtender Core FD	Köck
[2]		FAD SIP Server Provisioning & Network Management	Wiesinger / Gruber

[3]		SSH OCM Design	Cuijpers
[4]		NetXtender – SIP services for SIP server provisioning Manual	Cuijpers
[5]	2004-03-19	NetXtender - SNMP traps for SIP server provisioning	Cuijpers N.

1. OVERVIEW

This document describes the design and implementation of the services and atomic actions for SIP server provisioning. It is assumed that the reader of this document is familiar with at least [2] and [4].

Chapter 2 describes the classes, that are used by the services and chapter 3 describes all services and atomic actions.

2. Common classes

This chapter describes the classes that are used by the services:

- Class UUID, contain OID generation logic
- Class Constants, containing constant values
- Class ConfiguratorSIP, contains configuration logic and access rights logic

2.1. UUID class

The UUID class contains the method

```
public static String getUUID()
```

that generates values for the m_OID fields in the database. The UUID depends on the current time and some random numbers.

2.2. Constants class

The Constants class contains the constant values that are used in the SIP services and atomic actions. All variables in this class are defined as `public static final`.

2.3. ConfiguratorSIP class

The ConfiguratorSIP class contains 4 public methods: The method

```
public static void reloadConfig()
```

(re)loads the configuration from the configuration file.

The method

```
public static final String getConf(final String key)
```

reads a configuration setting from the cached configuration.

The method

```
public static final boolean checkAccess(final String key, String rule)
```

checks if a key(service) entry contains a certain rule(role). If the rule is available the method returns true, otherwise it returns false. For more information about configuration, see [4].

And the method

```
public static boolean checkPrivileges(String service, TiceData data)
```

which checks if the caller defined in the TiceData object is allowed to execute the service with the data defined in the TiceData object. First the role of the user is checked using the ConfiguratorSIP class. If the role of the caller is not allowed to execute the service, then the method returns false. Then all allowed roles from [2] are checked for each service.

3. SIP Services

This chapter describes the services and atomic actions that are implemented in the SIP server provisioning project. Section 3.1 contains a short overview of NTX services and atomic actions. Section 3.2 describes how schema-files are used to check the validity of the requests from client. Section 3.3 describes the CheckSystem service. Section 3.4 describes the DeleteTables atomic action which is used in some services. Section 3.5 describes the AbstractSIPService class and finally section 3.6 describes all other services and atomic actions for the SIP server provisioning project.

3.1. Overview

All services for NetXtender Light must be derived from the class ServiceLight. In this project, an abstract class 'AbstractSIPService' is introduced from which all services are derived. The AbstractSIPService class contains usefull methods that are used in nearly all services. See Figure 1 for a class diagram.

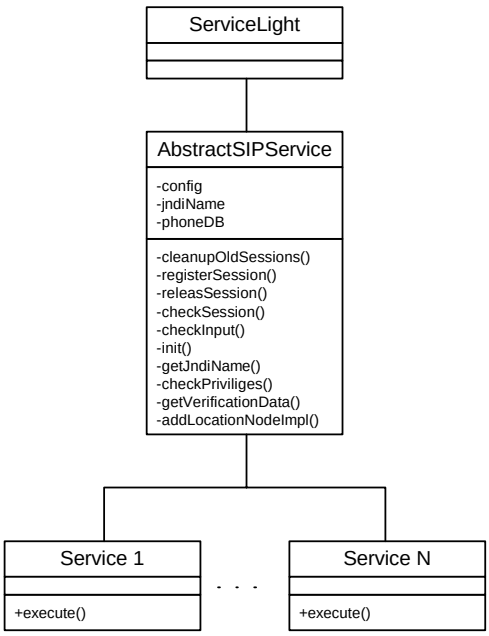


Figure 1 Class diagram of the services

All atomic actions for NetXtender Light must be derived from the class AtomicLight. See Figure 2 for a class diagram.

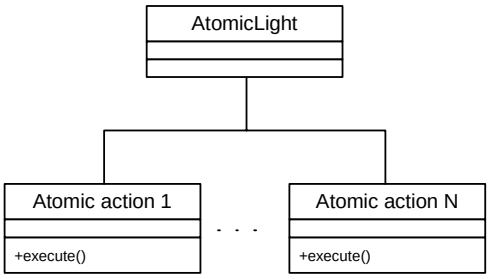


Figure 2 Class diagram of the atomic actions

3.2. Schemas

Schema files are used to check the input xml for the services. The schema files only check the contents of the data section. Figure 3 shows an example of a schema file.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <xsd:simpleType name="NonEmptyString">
    - <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
      </xsd:restriction>
    </xsd:simpleType>
  - <xsd:element name="data">
    - <xsd:complexType>
      - <xsd:sequence>
        - <xsd:element maxOccurs="1" minOccurs="1" name="Caller">
          - <xsd:complexType>
              <xsd:attribute name="uri" type="NonEmptyString" use="required" />
              <xsd:attribute name="session" type="NonEmptyString" use="required" />
            </xsd:complexType>
          </xsd:element>
        - <xsd:element maxOccurs="1" minOccurs="1" name="GroupImpl">
          - <xsd:complexType>
            - <xsd:sequence>
                <xsd:element maxOccurs="1" minOccurs="1" name="m_OID" type="NonEmptyString" />
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        - <xsd:element maxOccurs="1" minOccurs="1" name="Users">
          - <xsd:complexType>
            - <xsd:sequence>
              - <xsd:element maxOccurs="unbounded" minOccurs="0" name="UserImpl">
                - <xsd:complexType>
                  - <xsd:sequence>
                      <xsd:element maxOccurs="1" minOccurs="1" name="m_OID" type="NonEmptyString" />
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Figure 3 Schema example

The schema file shown in Figure 3 verifies that the input xml has a structure as shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <payload>
  - <header>
      <user-id>MusterMann</user-id>
      <trust-id>Kapsch</trust-id>
    </header>
  - <request service-id="net.kapsch.sip.service.AddUsersToLocalDomainGroup">
    - <data>
        <Caller uri="wiesinge@kapsch.net" session="1" />
      - <GroupImpl>
          <m_OID>003FD08B4E-F3BE690B-G2</m_OID>
        </GroupImpl>
      - <Users>
        - <UserImpl>
            <m_OID>003FD06DFA-34B2835E-User1</m_OID>
          </UserImpl>
        - <UserImpl>
            <m_OID>f072839e-cefb-d83c-6edc-041c74d7</m_OID>
          </UserImpl>
        </Users>
      </data>
    </request>
  </payload>
```

Figure 4 Example of a valid request described by the schema in Figure 3

The schema check is performed for each service. The name of the schema file must be the same as the name of the class file with extension. For the class

`net.kapsch.sip.service.AddUsersToLocalDomainGroup`

the schema file

`net.kapsch.sip.schema. AddUsersToLocalDomainGroup`

must be available.

3.3. CheckSystem service

The CheckSystem service checks if the NetXtender and the databases (sip and phone) are available. It uses the CheckDatabase atomic action to check the databases.

The data section of the CheckDatabase atomic action is empty. The CheckDatabase atomic action checks the database that is set in the 'destination' and throws a CommunicationException when the database is not available. The service catches the exception and creates the correct error message and throws a new CommunicationException.

No roles or access rights are defined for this service. This means that everybody can execute this service. No session is required to execute this service, so there is no need to be logged in.

3.4. DeleteTables atomic action

The atomic action `DeleteTables` is a specialized a.a., which is used by some -delete- services. The a.a. accepts the following input:

```
<TableList>
   <Table name = 'tableName_1' m_OID|m_parentOID='...'>
      <SubTable name = 'subTableName_1'/> (optional)
   </Table>
   <Table name = 'tableName_2' m_OID|m_parentOID='...'>
   </Table>
```

```
   ...
</TableList>
```

This input leads to a sequence of following delete statements:

> First entry:

1. `DELETE <subTableName_1> FROM <tableName_1>,<subTableName_1> WHERE <subTableName_1>.m_ParentOID=<tableName_1>.m_OID AND <tableName_1>.m_OID)= '...'`
2. `DELETE <tableName_1> FROM <tableName_1> WHERE <tableName_1>.(m_OID|m_ParentOID)= '...'`

> Second entry:

3. `DELETE <tableName_2> FROM <tableName_2> WHERE <tableName_2>.(m_OID|m_ParentOID)= '...'`

The 1:n relation between Subtable and table is implemented within the a.a. The Subtable element is optional in the input. The list may contain arbitrary table entries.

The following errors are thrown directly by the a.a.:

> DataExecption
Missing or inconsistent input data

3.5. AbstractSIPService

Since the SIP server provisioning server project includes many services, the use of an abstract class of which all services are derived, has the following benefits:

- Functionality's that are required in multiple services only need to be implemented once.
- Service code is shorter and therefore more readable.

This section describes which functionality's are implemented in the AbstractSIPService class. Nearly all services call a method called `init()` when the service starts executing. The init method performs four tasks which are described in sections to 3.5.1 to 3.5.4. Figure 5 shows a sequence diagram of the init method. Section 3.5.5 describes other methods that are implemented in the AbstractSIPService class.

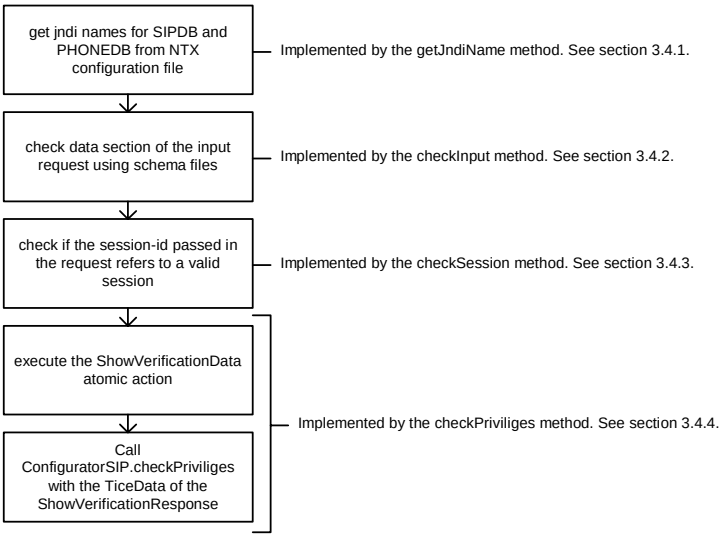


Figure 5 Sequence diagram of the init method

3.5.1. The getJndiName method

This method reads the following properties from the NetXtender configuration:

- xtender.datasource, the value defined in the properties is assigned to the local variable `jndiName`.
- xtender.datasourcePhone, the value defined in the properties is assigned to the local variable `phoneDB`.

If any value is not defined or the properties are not found a ConfigurationException is thrown.

3.5.2. The checkInput method

The checkInput method uses the schema files to check weather the data section of the request contains valid data. If the data section contains invalid data, then a DataException is thrown. If the configuration is invalid, then a ConfigurationException is thrown.

3.5.3. Session handling

Session handling is performed by four methods in the AbstractSIPService class:

- The `registerSession` method, which is only executed from within the Login request. It inserts a record into the sessionimpl database table containing the m_OID(session id), the users URI and the current time.
- The `checkSession` method is called by every request through the `init` method. It checks if the session id that was passed with the request, is valid and is not expired. Session expiration can be configured as described in [4]. If the session id is valid, then the time of the database is updated to the current time.

- The `releaseSession` method is only executed from within the Logout request. It checks if the session exists in the sessionimpl database table and deletes it. If the session can not be found in the database, then an XtenderException with error code `SECURITY_ERROR` is thrown.
- The `cleanupOldSessions` method is executed from within each Login service. It checks if there are old expired sessions to be deleted and deletes them. Session delete times can be configured as described in [4].

3.5.4. The checkPriviliges method

The last task of the init method is checking privileges. The `checkPriviliges` method executes the ShowVerificationData atomic action (see 3.5.4.1). The data section of the ShowVerificationData response is passed to the ConfiguratorSIP.checkPriviliges method which checks the privileges. If the caller does not have sufficient privileges to perform the service, then a DataException with error code `SECURITY_ERROR` is thrown.

3.5.4.1. The ShowVerificationData atomic action

The input for the ShowVerificationData atomic action is shown below. *Cursive* nodes are optional.

```
<request service-id="net.kapsch.sip.atomicaction.ShowVerificationData" >
   <data>

      <LocationImpl>
         <Attribute m_Key='JNDI1'  m_Value='...'>
         <Attribute m_Key='JNDI2'  m_Value='...'>
         ...
      </LocationImpl>

      <Caller uri="... "/>

      <UserImpl>
         <m_SIPURI>...</m_SIPURI>              OR       <m_OID>...</m_OID>
        <m_ResetPWD>
      </UserImpl>

      <GroupImpl>
         <m_OID>...</m_OID>                    OR       <m_GroupType>domain|local</m_OID>
      </GroupImpl>

      <DomainImpl>
         <m_DomainName>...</m_ DomainName >    OR       <m_OID>...</m_OID>
      </DomainImpl>

   </data>
</request>
```

The LocationImpl node is mandatory and requires at least one Attribute node. Attribute nodes define databases. The ShowVerificationData atomic action tries all databases specified here. If all fail, then an exception is thrown. Otherwise, the working database JNDI name is returned in the response.

The caller node is mandatory.

The UserImpl, GroupImpl and DomainImpl nodes are optional. If one of these nodes is passed with a request, then a node will be available in the response.

If a UserImpl node a m_SIPURI is specified, then the data as shown in the response below corresponding to the m_SIPURI is returned. Otherwise, if the m_OID is specified, the data corresponding to the m_OID is returned.

If a DomainImpl node a m_DomainName is specified, then the data as shown in the response below corresponding to the m_DomainName is returned. Otherwise, if the m_OID is specified, the data corresponding to the m_OID is returned.

The <CustomerAdmin> node is returned, in case the <resetPWD> node is present and the User has not Email address set.

The output of the ShowVerificationData atomic action is shown below:

```
<response>
   <responseCode identifier="SUCCESS|FAILED"/>
   <data>

      <LocationImpl>
         <Attribute m_Key="JNDI" m_Value="..."/>"
      </LocationImpl>

      <CallerImpl>
         <m_OID>003FB3738B-2FCF2A97-566A6D15</m_OID>
         <m_Type>User</m_Type>
         <m_SIPURI>wiesinge@kapsch.net</m_SIPURI>
         <m_DomainOID>kapsch.net</m_DomainOID>
         <m_Role>...</m_Role>
      </CallerImpl>

      <UserImpl> (optional)
         <m_OID>003FB3738B-2FCF2A97-566A6D15</m_OID>
         <m_Type>User</m_Type>
         <m_SIPURI>wiesinge@kapsch.net</m_SIPURI>
         <m_DomainOID>kapsch.net</m_DomainOID>
         <m_Email>…</m_EMail>
         <m_Role>...</m_Role>
      </UserImpl>
      <UserImplCustomerAdmin> (optional)
         <m_OID>003FB3738B-2FCF2A97-566A6D15</m_OID>
         <m_Type>User</m_Type>
         <m_SIPURI>wiesinge@kapsch.net</m_SIPURI>
         <m_DomainOID>kapsch.net</m_DomainOID>
         <m_Email>…</m_EMail>
         <m_Role>...</m_Role>
      </UserImplCustomerAdmin>

      <GroupImpl> (optional)
         <m_OID>003FB3738B-2FCF2A97-566A6D15</m_OID>
         <m_DomainOID>...</ m_DomainOID>
         <m_GroupType>...</ m_ GroupType>
      </GroupImpl>

      <DomainImpl> (optional)
         <m_OID>...</ m_OID >
         <m_DomainName>...</ m_DomainName >
      </DomainGroupImpl>

   </data>
</response>
```

3.5.5. Other methods

3.5.5.1. GetVerificationData method

The getVerificationData method can be used by each service to get access to the response of the ShowVerificationData atomic action.

3.5.5.2. AddLocationImplNode method

The `addLocationImplNode` method adds the following xml structure to a TiceData object:

```
<LocationImpl>
   <Attribute m_Key="JNDI" m_Value="jndiName">
</LocationImpl>
```

where jndiName is passed as argument.

3.5.5.3. GenerateTrap

This method is a wrapper method for sending an SMP Trap. For further details concerning the SNMP messaging please see [5].

3.5.5.4. SendSMTPMessage

This method is a wrapper method for sending an Email notification. Receicer, Sender, subject and message text are input data, the rest (Email HOST) is configured in the log4j.xml configuration file of the JBoss server.

3.6. Common services functionality

All services (except in Login and CheckSystem) call the `init()` method at the start of execution (the functionality of the `init()` method already described in section 3.5)

The Login service only calls the following methods:

1. `getJndiName()`
2. `checkInput()`
3. `checkPrivileges()`

Because during login no session has to be checked, because no session exists.

The CheckSystem only calls the getJndiName() method to get the database's jndi name.

Before an atomic action is started, the services always call the `addLocationImplNode` method described in section 3.5.5.2.

Before returning the response to the client, all services (except the CheckSystem service) add the 'Caller' node from the request to the data section of the response.

3.6.1. Service GenericDBService

3.6.1.1. Functionality

The generic service (implemented for package `install` and `profile`) is used as a dispatcher service for SIP simple services without extra logic. The functionality depends on the fact, that the original service-id within the incoming TICE request is transported in the data section for evaluation. Flow of control:

> Original service-id is retrieved from data section
> The init method of the AbstractSipService class is called – which:
  > retrieves configuration, check input against schema, verifies session, checks privileges (throws exceptions in case of error)
> Depending on the service, Check – actions are called
> The a.a. for the original service is called – the name is build:
  > mapped hard coded
  > generated out of the original service name
> The a.a. is called and the response is returned

Login service executes the ShowUserDetailedData atomic action (see section 3.6.14.2) to get the data to check if the user exists and if the password is correct. If the user does not exist, then a DataException with error code LOGIC_ERROR is thrown.

3.6.2. Service Login

3.6.2.1. Functionality

The Login service executes the ShowUserDetailedData atomic action (see section 3.6.14.2) to get the data to check if the user exists and if the password is correct. If the user does not exist, then a DataException with error code LOGIC_ERROR is thrown.

It then checks, whether the Authentication Failure check is active (parameter in SIP_Conf.xml). This check limits the number of failed logins due to a wrong password. For each user, an authentication failure counter is maintained (userimpl→featureimpl→attributeimpl). The number of unsuccessful logins is limited for each domain (userimpl→domainimpl→attributeimpl). For each unsuccessful login, the counter is incremented, for each successful login, the counter is resetted. If the user has reached the maximum number of unsuccessful logins, he is blocked. In this situation, his counter can only be resetted by the service ResetPassword.

- A wrong password is indicated by the ResponseCode SUCCESS and the ResponseCode message SEC_AUTHORIZATION: Wrong password (This avoids rolling back the service and also rolling back the incrementing of the counter!).
- All other errors are indicated by a SecurityException with error code SEC_AUTHORIZATION.

If the user exists and the password is correct, then a new session for the user is registered using the registerSession method described in 3.5.3.The response contains the output of the ShowUserDetailedData atomic action except for the password node. The caller node of the request is added to the response with the new session-id that was generated by the registerSession method. The user has to send the session-id with all following requests.

3.6.3. Service Logout

3.6.3.1. Functionality

The Logout service releases the user's session using the releaseSession method described in 3.5.3. If the user does not have a session with the corresponding session-id, then an XtenderException with error code SECURITY_ERROR is thrown.

3.6.4. Service ShowFeaturesForUser

3.6.4.1. Functionality

The ShowFeaturesForUser service executes the ShowFeaturesForUser atomic action and returns the result of that atomic action.

3.6.4.2. Atomic action ShowFeaturesForUser

The ShowFeaturesForUser atomic action queries all features (or optionally one feature defined in the input) from the user and their attributes from the featureimpl and attributeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key=”JNDI” m_Value=”...” />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <FeatueImpl>
    <FeatureName>…</FeatureName>          optional
   </FeatureImpl>
</data>
```

The response is

```
<data>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Type>...</m_Type>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_OID = “...” m_Type=”...” m_Key=”...” m_Value=”...” />
         <Attribute m_OID = “...” m_Type=”...” m_Key=”...” m_Value=”...” />
         ...
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Type>...</m_Type>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>.</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_OID = “...” m_Type=”...” m_Key=”...” m_Value=”...” />
         <Attribute m_OID = “...” m_Type=”...” m_Key=”...” m_Value=”...” />
         ...
      </FeatureImpl>
      ...
   </Features>
</data>
```

3.6.5. Service AddFeaturesForUser

3.6.5.1. Functionality

The AddFeaturesForUser service executes the AddFeaturesForUser atomic action and returns the result of the atomic action.

3.6.5.2. Atomic action AddFeaturesForUser

The AddFeaturesForUser atomic actions inserts features and their attributes into the featureimpl and attributeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <Features>
      <FeatureImpl>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_Key="..." m_Value="..." />
         <Attribute m_Key="..." m_Value="..." />
         ...
      </FeatureImpl>
      <FeatureImpl>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_Key="..." m_Value="..." />
         <Attribute m_Key="..." m_Value="..." />
         ...
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.6. Service ModifyFeaturesForUser

3.6.6.1. Functionality

The ModifyFeaturesForUser service checks if the features and attributes belong to the user. If one of the attributes or features do not belong to the user, then a LogicalException with error code LOGIC_ERROR is thrown. Otherwise, the ModifyFeaturesForUser atomic action is executed.

3.6.6.2. Atomic action ModifyFeaturesForUser

The ModifyFeaturesForUser atomic actions updates features and their attributes of the featureimpl and attributeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <Features>
```

```
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attributes>
            <Add>    <!-- new attributes -->
               <Attribute m_Key="..." m_Value="..." />
               <Attribute m_Key="..." m_Value="..." />
               ...
            </Add>
            <Modify> <!-- change these existing attributes -->
               <Attribute m_OID="..." m_Key="..." m_Value="..." />
               <Attribute m_OID="..." m_Key="..." m_Value="..." />
               ...
            </Modify>
            <Delete> <!-- delete these attributes -->
               <Attribute m_OID="..." />
               <Attribute m_OID="..." />
               ...
            </Delete>
         </Attributes>
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attributes>
            <Add>    <!-- new attributes -->
               <Attribute m_Key="..." m_Value="..." />
               <Attribute m_Key="..." m_Value="..." />
               ...
            </Add>
            <Modify> <!-- change these existing attributes -->
               <Attribute m_OID="..." m_Key="..." m_Value="..." />
               <Attribute m_OID="..." m_Key="..." m_Value="..." />
               ...
            </Modify>
            <Delete> <!-- delete these attributes -->
               <Attribute m_OID="..." />
               <Attribute m_OID="..." />
               ...
            </Delete>
         </Attributes>
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.7. Service ModifyUserDetailedData

3.6.7.1. Functionality

The ModifyUserDetailed calls the ModifyUserDetailed atomic action. It then calls service ShowUserDetailedData and returns the updated data for the user.

3.6.7.2. Atomic action ModifyUserDetailedData

The ModifyUserDetailedData atomic actions updates columns m_FirstName, m_LastName, m_Mobile, m_Email in the userimpl table. The input of the atomic action is

```
<data>
        <Caller uri="2@gg.at" session="1"/>
        <UserImpl>
          <m_OID>25047b83-bcff-37db-cfb3-c4fec89c0ce7</m_OID>
```

```
            <m_FirstName>Albert</m_FirstName>
            <m_LastName>Einstein</m_LastName>
            <m_Mobile>+43664XXXXX</m_Mobile>
            <m_Email>albert.einstein@genie.net</m_Email>
            <m_Role>philosoph</m_Role>
          </UserImpl>
    </data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.8. Service ChangeFeatureStatus

3.6.8.1. Functionality

The ChangeFeatureStatus service checks if the features belong to the user. If one of the features does not belong to the user, then a LogicalException with error code LOGIC_ERROR is thrown. Otherwise, the ChangeFeatureStatus atomic action is executed.

3.6.8.2. Atomic action ChangeFeatureStatus

The ChangeFeatureStatus atomic action updates the m_Status field of the featureimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Status>...</m_Status>
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Status>...</m_Status>
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.9. Service DeleteFeaturesForUser

3.6.9.1. Functionality

The DeleteFeaturesForUser service checks if the features belong to the user. If one of the features does not belong to the user, then a LogicalException with error code LOGIC_ERROR is thrown. Otherwise, the DeleteFeaturesForUser atomic action is executed.

3.6.9.2. Atomic action DeleteFeaturesForUser

The DeleteFeaturesForUser atomic deletes records from the featureimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
```

```
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.10. Service CreateUser

3.6.10.1. Functionality

The CreateUser service executes the CreateUser atomic action and then the AddFeaturesForUser atomic action (see 3.6.5.2).

3.6.10.2. Atomic action CreateUser

The CreateUser atomic action checks if the MainGroupFeatures passed in the request belongs to the domain of the user and if it is of the correct type. Then it checks if the MainGroupRouting passed in the request belongs to the domain of the user and if it is of the correct type. If one of the conditions is false, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the user data is inserted into the userimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_FirstName>...</m_FirstName>
      <m_LastName>...</m_LastName>
      <m_Password>...</m_Password>
      <m_Password_Encrypted>...</m_Password_Encrypted>
      <m_Mobile>...</m_Mobile>
      <m_Email>...</m_Email>
      <m_DomainOID>...</m_DomainOID>
      <m_MainGroupOIDRouting>...</m_MainGroupOIDRouting>
      <m_MainGroupOIDFeatures>...</m_MainGroupOIDFeatures>
      <m_PrimaryLocation>...</m_PrimaryLocation>
      <m_GatewayLocation>...</m_ GatewayLocation>
      <m_Role>...</m_Role>
   </UserImpl>
</data>
```

The output of the atomic action contains the oid of the new user:

```
<data>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
</data>
```

3.6.11. Service DeleteUser

3.6.11.1. Functionality

The DeleteUser service deletes a user from the database by executing the DeleteUser atomic action.

3.6.11.2. Atomic action DeleteUser

The DeleteUser atomic action deletes a user from the userimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.12. Service AddAlias

3.6.12.1. Functionality

The AddAlias service executes the AddAlias atomic action.

3.6.12.2. Atomic action AddAlias

The AddAlias atomic action inserts a record into the aliasimpl database table. The input for the atomic action is

```
<data>
 <LocationImpl>
    <Attribute m_Key="JNDI" m_Value="..." />
 </LocationImpl>
 <UserImpl>
    <m_OID>...</m_OID>
 </UserImpl>
 <AliasImpl>
    <m_URI>...</m_URI>
    <m_Password>...</m_Password>
    <m_Password_Encrypted>...</m_Password_Encrypted>
 </AliasImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.13. Service DeleteAlias

3.6.13.1. Functionality

The DeleteAlias service deletes an alias of a user by executing the DeleteAlias atomic action

3.6.13.2. Atomic action DeleteAlias

The DeleteAlias atomic action deletes the corresponding record from the aliasimpl database table. If the alias to delete does not exist, then a LogicalException with error code DATA_FAULT is thrown. The input for the atomic action is

```
<data>
 <LocationImpl>
    <Attribute m_Key="JNDI" m_Value="..." />
 </LocationImpl>
 <UserImpl>
    <m_OID>...</m_OID>
 </UserImpl>
 <AliasImpl>
    <m_URI>...</m_URI>
 </AliasImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.14. Service ShowUserDetailedData

3.6.14.1. Functionality

The ShowUserDetailedData service first executes the ShowUserDetailedData atomic action and then it executes the ShowPhoneData atomic action. Then it combines the responses of the two atomic action to the service response as described in [2].

3.6.14.2. Atomic action ShowUserDetailedData

The ShowUserDetailedData atomic action queries the user data from the database. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <key id="OID | URI" value="..." />
</data>
```

The atomic action can retrieve the data for either a user oid or a user's uri. It also selects all rows of the connected table contactimpl (userimpl→contactimpl) and returns them in the response

The response of the atomic actions is

```
<data>
   <UserImpl>
      <m_OID>...</m_OID>
      <m_Type>User</m_Type>
      <m_SIPURI>...</m_SIPURI>
      <m_Password>...</m_Password>
      <m_Password_Encrypted>...</m_Password_Encrypted>
      <m_FirstName>...</m_FirstName>
      <m_LastName>...</m_LastName>
      <m_Mobile>...</m_Mobile>
      <m_Email>...</m_Email>
      <m_WatcherInfo>...<m_WatcherInfo>
      <m_DomainOID>...</m_DomainOID>
      <m_DomainName>...</m_DomainName>
      <m_MainGroupOIDRouting>...</m_MainGroupOIDRouting>
      <m_MainGroupOIDFeature>...</m_MainGroupOIDFeature>
      <m_PrimaryLocation>...</m_PrimaryLocation>
   <m_GatewayLocation>...</m_ GatewayLocation>
   </UserImpl>
   <Role id = "..." />
```

```
   <LocationImpl>
      <m_Location>...</m_Location>
      <Attribute m_Key="..." m_Value="..." />
      <Attribute m_Key="..." m_Value="..." />
      ...
   </LocationImpl>
   <Aliases>
      <AliasImpl>
         <m_URI>...</m_URI>
         <m_Password_Encrypted>...</m_Password_Encrypted>
      </AliasImpl>
   </Aliases>
   <Groups>
      <GroupImpl referenced-by="GroupMappingImpl">
         <m_GroupName>...</m_GroupName>
      </GroupImpl>
   </Groups>
</data>
```

3.6.14.3. Atomic action ShowPhoneData

The ShowPhoneData atomic action reads the phone data from the database. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_SIPURI>...</m_SIPURI>
   </UserImpl>
</data>
```

The output of the atomic action is

```
<data>
   <Phone_Attr>
      <MAC_Addr>...</MAC_Addr>
      <phone_type>...</phonne_type>
   <Phone_Attr>
</data>
```

or

```
<data/>
```

when no phone data is found in the database.

3.6.15. Service ChangePassword

3.6.15.1. Functionality

The ChangePassword service executes the ChangePassword atomic action.

3.6.15.2. Atomic action ChangePassword

The ChangePassword atomic action queries the old password from the userimpl database table and compares it with the old password that was passed with the request. If the user can not be found or the passwords do not match, then a LogicalException is thrown with error code LOGIC_ERROR. Otherwise, the password in the database is updated. The input of the atomic action is

```
   <data>
      <LocationImpl>
         <Attribute m_Key="JNDI" m_Value="..." />
      </LocationImpl>
      <UserImpl>
         <m_OID>...</m_OID>
```

```
      <m_OldPassword>...</m_OldPassword>
      <m_Password>...</m_Password>
      <m_Password_Encrypted>...</m_Password_Encrypted>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.16. Service ChangeAliasPassword

3.6.16.1. Functionality

The ChangeAliasPassword service executes the ChangeAliasPassword atomic action.

3.6.16.2. Atomic action ChangeAliasPassword

The ChangeAliasPassword atomic action checks if the alias exists in the database. If the alias does not exist, then a LogicalException is with error code LOGIC_ERROR is thrown. Then the old password that was passed with the request is checked with the password in the database. If the passwords do not match, then a LogicalException is thrown with error code LOGIC_ERROR. Otherwise, the password in the database is updated. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <AliasImpl>
      <m_URI>...</m_URI>
      <m_OldPassword>...</m_OldPassword>
      <m_Password>...</m_Password>
      <m_Password_Encrypted>...</m_Password_Encrypted>
   </AliasImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.17. Service ResetPassword

3.6.17.1. Functionality

The service checks, wheter the caller is entitled to carry out this service.

It then checks, whether the Authentication Failure check is active (parameter in SIP_Conf.xml). If active, the authentication failure counter for the user is reset in the SIP DB (userimpl→featureimpl→attributeimpl) by calling the atomic actions ShowFeaturesForUser and ModifyFeaturesForUser.

After that, the ResetPassword service executes the ResetPassword atomic action. Then an Email notification is sent. The sender is taken from the email address of the caller, the receiver is taken from the Email address of the user. If there is no Email address configured for the user, the Email address of his customer Domain admin is taken. The emai address has a fixed subject and message text.

3.6.17.2. Atomic action ResetPassword

The ResetPassword atomic action updates the password in the userimpl database table. If the user can not be found, then a LogicalException is with error code LOGIC_ERROR is thrown. The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
      <m_Password>...</m_Password>
      <m_Password_Encrypted>...</m_Password_Encrypted>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.18. Service ResetAliasPassword

3.6.18.1. Functionality

The ResetAliasPassword service executes the ResetAliasPassword atomic action.

3.6.18.2. Atomic action ResetAliasPassword

The ResetAliasPassword atomic action clears the password in the aliasimpl database table. If the user can not be found, then a LogicalException is with error code LOGIC_ERROR is thrown. The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.19. Service ChangeUserMainRoutingGRPID

3.6.19.1. Functionality

The ChangeUserMainRoutingGRPID Service checks if the grouptype that was returned by the ShowVerificationData (see 3.5.4.1) atomic action matches `Constants.GROUP_TYPE_DOMAIN_ROUTNG`. If they do not match, then a LogicalException with error code LOGIC_ERROR is thrown. Then the service executes the ChangeUserMainRoutingGRPID atomic action.

3.6.19.2. Atomic action ChangeUserMainRoutingGRPID

The ChangeUserMainRoutingGRPID atomic action updates the m_MainGroupOIDRouting field of the userimpl database table. When the user can not be found, then a LogicalException with error code LOGIC_ERROR is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
```

```
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
      <m_MainGroupOIDRouting>...</m_MainGroupOIDRouting>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.20. Service ChangeUserMainGRPFeature

3.6.20.1. Functionality

The ChangeUserMainGRPFeature Service checks if the grouptype that was returned by the ShowVerificationData (see section 3.5.4.1) atomic action matches `Constants.GROUP_TYPE_DOMAIN_FEATURES`. If they do not match, then a LogicalException with error code LOGIC_ERROR is thrown. Then the service executes the ChangeUserMainGRPFeature atomic action.

3.6.20.2. Atomic action ChangeUserMainGRPFeature

The ChangeUserMainGRPFeature atomic action updates the m_MainGroupOIDFeatures field of the userimpl database table. When the user can not be found, then a LogicalException with error code LOGIC_ERROR is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..." />
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
      <m_MainGroupOIDFeature></m_MainGroupOIDFeature>
   </UserImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.21. Service QueryBuddyList

3.6.21.1. Functionality

The QueryBuddyList service executes the ShowUserDetailedData atomic action (see 3.6.14.2) to get the location of the user. Then the QueryBuddyList atomic action is executed.

3.6.21.2. Atomic action QueryBuddyList

The QueryBuddyList atomic action tries to find a database with the jndi names from the request. If non of the databases are available, then a DatabaseException with error code `DB_FAULT_DRIVER` is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI1" m_Value="..." />
      <Attribute m_Key="JNDI2" m_Value="..." />
      ...
   </LocationImpl>
   <UserImpl>
      <m_OID>...</m_OID>
```

```
      </UserImpl>
   </data>
```

The output of the atomic action is

```
<data>
   <BuddyList>
      <BuddyListImpl>
         <m_OID>...</m_OID>
         <m_Type>BuddyList</m_Type>
         <UserImpl>
            <m_OID>...</m_OID>
            <m_Type>User</m_Type>
            <m_SIPURI>...</m_SIPURI>
            <m_Password>...</m_Password>
            <m_Password_Encrypted>...</m_Password_Encrypted>
            <m_FirstName>...</m_FirstName>
            <m_LastName>...</m_LastName>
            <m_Mobile>...</m_Mobile>
            <m_Email>...</m_Email>
            <m_Status>...<m_Status>
            <m_PrimaryLocation>...</m_PrimaryLocation>
         </UserImpl>
      </BuddyListImpl>
      <BuddyListImpl>
         <m_OID>...</m_OID>
         <m_Type>BuddyList</m_Type>
         <UserImpl>
            <m_OID>...</m_OID>
            <m_Type>User</m_Type>
            <m_SIPURI>...</m_SIPURI>
            <m_Password>...</m_Password>
            <m_Password_Encrypted>...</m_Password_Encrypted>
            <m_FirstName>...</m_FirstName>
            <m_LastName>...</m_LastName>
            <m_Mobile>...</m_Mobile>
            <m_Email>...</m_Email>
            <m_Status>...<m_Status>
            <m_PrimaryLocation>...</m_PrimaryLocation>
         </UserImpl>
      </BuddyListImpl>
      ...
   </BuddyList>
</data>
```

3.6.22. Service SearchUser

3.6.22.1. Functionality

The SearchUser service executes the SearchUser atomic action.

3.6.22.2. Atomic action SearchUser

The SearchUser atomic action queries the database for users. Search criteria can be a combination of URI, first name and last name. The input of the atomic action is

```
   <data>
      <LocationImpl>
         <Attribute m_Key=”JNDI” m_Value=”...” />
      </LocationImpl>
      <Caller uri=”...” />
      <UserImpl>
         <m_SIPURI>...</m_SIPURI>
         <m_FirstName>...</m_ FirstName>
         <m_ LastName>...</m_LastName>
      </UserImpl
   </data>
```

The response of the atomic action is

```
<data>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_SIPURI>...</m_SIPURI>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_DomainOID>...</m_DomainOID>
         <m_DomainName>...</m_DomainName>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_SIPURI>...</m_SIPURI>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_DomainOID>...</m_DomainOID>
         <m_DomainName>...</m_DomainName>
      </UserImpl>
      ...
   </Users>
</data>
```

3.6.23. Service SetVoicePromptPathForDomain

3.6.23.1. Functionality

The SetVoicePromptPathForDomain service:

> inserts one row/attribute in table `attributeimpl` (a.a. `AddAttributesForDomain`) for the given Domain
> calls the script `CreateDomainAnncDirectory` via an SSH call on the local UNIX host.

Location of the script can be configured in the config file SIP_config.xml.

3.6.24. Service CreateVoiceMailDirectoryForUser

3.6.24.1. Functionality

The CreateVoiceMailDirectoryForUser service:

> calls the script `CreateVoiceMailDirectoryForUser` via an SSH call on the local UNIX host.
  > Invocation parameters: --m_SIPURI –sip_db_user –sip_db_pwd

Location of the script can be configured in the config file SIP_config.xml.

3.6.25. Abstract Atomic action AbstractSIPAtomicAction

The SIP platform supports a licensing model. For each domain, a maximum number of allowed users is defined (Attributeimpl entry `MaxRegisteredUsers`). If a domain is to be created, or this attribute is to be changed for an existing domain, the following steps are carried out:

2. SUM(m_ProxyMaxConcurrentCalls) of all locations found
3. SUM(MaxRegisteredUsers) of all domains except the domain, which shall be created/modified.

4. MaxRegisteredUser from input

If (2. + 3.) > 1.) : creation/modification is denied.

The licensing check can be set active/inactive using a configuration parameter of the SIP_Conf.xml.

The licensing check has impact on services CreateDomain and ModifyDomainDetailedData.

3.6.26. Service AddAttributesForDomain

3.6.26.1. Functionality

The AddAttributesForDomain service executes the `AddAttributesForDomain` atomic action.

3.6.26.2. Atomic action AddAttributesForDomain

Inserts one row for each `<AttributeImpl>` element of the input XML in table `attributeimpl`. The attributes belong to a input domain.

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <Attributes>
      <AttributeImpl>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      <AttributeImpl>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      ...
   </Attributes>
</data>
```

The response of the atomic action is

```
<data>
</data>
```

3.6.27. Service DeleteAttributesForDomain

3.6.27.1. Functionality

The DeleteAttributesForDomain service executes the `DeleteAttributesForDomain` atomic action.

3.6.27.2. Atomic action DeleteAttributesForDomain

Deletes one row for each `<AttributeImpl>` element of the input XML in table `attributeimpl`. The attributes belong to a input domain.

```
<data>
        <Caller uri="2@gg.at" session="1"/>
        <DomainImpl>
          <m_OID>2501d8ef-aefe-4f06-e7c3-c4fec89c0ce7</m_OID>
        </DomainImpl>
```

```
      <Attributes>
        <AttributeImpl>
          <m_OID>8a3d64df-81f2-3eba-767e-be38e0a2db59</m_OID>
        </AttributeImpl>
        <AttributeImpl>
          <m_OID>8a3d64e2-139f-64a8-3d7e-be38e0a2db59</m_OID>
        </AttributeImpl>
      </Attributes>
    </data>
```

The response of the atomic action is

```
<data>
</data>
```

3.6.28. Service CreateDomain

3.6.28.1. Functionality

The CreateDomain service executes the `CreateDomain` and optionally (depending on the input) the `AddGatewayLocationsOfDomain` atomic action.

3.6.28.2. Atomic action CreateDomain

The CreateDomain atomic action calls checkMaxRegisteredUser of the derived abstract class (throws exception in case the domain must not be created) and inserts a record in the domainimpl database table and zero or more records in the attributeimpl database table. The input of the atomic action is

```
   <data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m DomainName>...</m DomainName>
      <m_Description>...</m_Description>
   </DomainImpl>
   <Attributes>
      <AttributeImpl>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      <AttributeImpl>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      ...
   </Attributes>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

3.6.29. Service DeleteDomain

3.6.29.1. Functionality

The DeleteDomain service executes the following modules sequentially:

- `A.A. DeleteDomain`: Deletes the domainimpl + userimpl+all referenced entries in the SIP Db
- `A.A. DeleteTables`: Deletes entries in PHONE_DB for table domain_attr and phone_attr
- `Service ExecutePhoneService`: Calls perls script ClearDomain on the default SSH Host.

3.6.29.2. Atomic action DeleteDomain

First, all users of the domain are deleted using the A.A: deleteUser (This a.a. also deletes featureimpl entries).

Second, all groups of the domain are deleted using the A.A. deleteLocalGroup (deletes also featureimpl, routingruleimpl entries).

Finally, the domain and its attributes, routingrules is deleted.

The input of the atomic action is

```
   <data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

The response of the atomic action is an empty <data/> section.

3.6.30. Service AddRoutingRuleForDomain

3.6.30.1. Functionality

The AddRoutingRuleForDomain service executes the AddRoutingRuleForDomain atomic action.

3.6.30.2. Atomic action AddRoutingRuleForDomain

The AddRoutingRuleForDomain atomic action inserts one record into the routingruleimpl database table. The atomic action maintains the order as defined in the database. For example: if two records exist in the database with order=1 and order=2, and a new record is to be inserted at position 2, then the record that was originally at order=2 will move to order=3. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If a too high or too low order is specified, then a LogicalException with error code DATA_FAULT is thrown. Also, if the domain can not be found in the database, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <RoutingRuleImpl>
      <m_Type>...</m_Type>
      <m_Order>...</m_Order>
      <m_Action>...</m_Action>
      <m_PatternFrom>...</m_PatternFrom>
```

```
      <m_PatternTo>....</m_PatternTo>
      <m_PatternRequestUri>
         <![CDATA[...]]>
      </m_PatternRequestUri>
      <m_Attribute>...</m_Attribute>
      <m_RequeryResult>...</m_RequeryResult>
      <m_FallThroughOnFail>...</m_FallThroughOnFail>
      <m_NightMode>...</m_NightMode>
      <m_TerminatingRule>...</m_TerminatingRule>
   </RoutingRuleImpl>
</data>
```

The response of the atomic action is:

```
<data>
   <RoutingRuleImpl>
      <m_OID>...</m_OID>
   </RoutingRuleImpl>
</data>
```

3.6.31. Service AddGatewayLocationsForDomain

3.6.31.1. Functionality

The AddGatewayLocationsForDomain service adds gateway location entries for a domain in the SIP DB. It executes the `AddGatewayLocationsForDomain` atomic action.

3.6.31.2. Atomic action AddGatewayLocationsForDomain

The `AddGatewayLocationsForDomain` atomic action inserts one record in table gatewayimpl for each gateway in the list. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <Gateways>
      <GatewayLocationImpl>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
      <GatewayLocationImpl>
      <GatewayLocationImpl>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
      <GatewayLocationImpl>
   </Gateways
</data>
```

The response of the atomic action is a empty data section:

```
</data>
```

3.6.32. Service ChangeGatewayLocationsForDomain

3.6.32.1. Functionality

The ChangeGatewayLocationsForDomain service changes gateway location entries for a domain in the SIP DB. It executes the `ChangeGatewayLocationsForDomain` atomic action.

3.6.32.2. Atomic action ChangeGatewayLocationsForDomain

The `ChangeGatewayLocationsForDomain` atomic action modifies one record in table gatewayimpl for each gateway in the list. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <Gateways>
      <GatewayLocationImpl>
         <m_OID>...</m_OID>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
      <GatewayLocationImpl>
      <GatewayLocationImpl>
         <m_OID>...</m_OID>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
      <GatewayLocationImpl>
   </Gateways
</data>
```

The response of the atomic action is a empty data section:

```
</data>
```

3.6.33. Service ChangeGatewayLocationForUser

3.6.33.1. Functionality

The ChangeGatewayLocationsForUser service changes the gateway location entry for a user in the SIP DB. It executes the `ChangeGatewayLocationsForUser` atomic action. The new gateway location must be in the same domain as the user.

3.6.33.2. Atomic action ChangeGatewayLocationsForUser

The `ChangeGatewayLocationsForDomain` atomic action modifies one record in table gatewayimpl for each gateway in the list. The input of the atomic action is

```
<data>
   <Caller uri="customer@kcc.at" session="1"/>
   <UserImpl>
      <m_OID>...</m_OID>
   </UserImpl>
   <GatewayLocationImpl>
      <m_OID>...</m_OID>
  </GatewayLocationImpl>
</data>
```

The response of the atomic action is a empty data section:

```
</data>
```

3.6.34. Service DeleteGatewayLocationsForDomain

3.6.34.1. Functionality

The DeleteGatewayLocationsForDomain service deletes gateway location entries for a domain in the SIP DB. It executes the `DeleteGatewayLocationsForDomain` atomic action.

3.6.34.2. Atomic action DeleteGatewayLocationsForDomain

The `DeleteGatewayLocationsForDomain` atomic action deletes one record in table gatewayimpl for each gateway in the list. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <Gateways>
      <GatewayLocationImpl>
         <m_OID>...</m_OID>
      <GatewayLocationImpl>
      <GatewayLocationImpl>
         <m_OID>...</m_OID>
      <GatewayLocationImpl>
   </Gateways
</data>
```

The response of the atomic action is a empty data section:

```
</data>
```

3.6.35. Service ShowGatewayLocationsForDomain

3.6.35.1. Functionality

The ShowGatewayLocationsForDomain service shows all gateway location entries for a domain in the SIP DB. The domain is optional:

> DomainImpl is set in input
> all gateway location entries of the given domain are shown
> DomainImpl is not set in input
> all gateway location entries of all domains are shown

The service calls the `ShowGatewayLocationsForDomain` atomic action.

3.6.35.2. Atomic action ShowGatewayLocationsForDomain

The `ShowGatewayLocationsForDomain` atomic action shows all records in table gatewayimpl for a given domain. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>                          optional
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

The response of the atomic action is:

```
<data>
   <Gateways>
      <GatewayLocationImpl>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
         <m_DomainOID>...</m_ DomainOID>
         <m_DomainName>...</m_ DomainName>
      <GatewayLocationImpl>
      <GatewayLocationImpl>
         <m_HeadNumber>...</m_HeadNumber>
         <m_Gateway>...</m_Gateway>
         <m_TrunkGrpID>...</m_TrunkGrpID>
         <m_DomainOID>...</m_ DomainOID>
         <m_DomainName>...</m_ DomainName>
      <GatewayLocationImpl>
   </Gateways
</data>
```

3.6.36. Service DeleteRoutingRuleForDomain

3.6.36.1. Functionality

The DeleteRoutingRulesForDomain service executes the DeletesRoutingRuleForDomain atomic action.

3.6.36.2. Atomic action DeleteRoutingRuleForDomain

The DeleteRoutingRuleForDomain atomic action removes records from the routingruleimpl database table. The atomic action maintains the order as defined in the database. For example: if two records exist in the database with order=1 and order=2, and the record with order=1 is to be removed, then the record that was originally at order=2 will move to order=1. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If a too high or too low order is specified, then a LogicalException with error code DATA_FAULT is thrown. Also, if the routing rule can not be found in the database, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <RoutingRules>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
      </RoutingRuleImpl>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
      </RoutingRuleImpl>
      ...
   </RoutingRules>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.36.3. Service ListDomains

3.6.36.4. Functionality

The ListDomains service executes the ListDomains atomic action.

3.6.36.5. Atomic action ListDomains

The ListDomains atomic action lists all domains that match a search criteria where wildcards are allowed. If an empty search criteria is specified, then all domains are listed.

Depending on the role of the caller, all domains or only the domain of the caller are returned. The following roles see:

> Role customeradmin: only his own domain
> Role callcenteragent, provideradmin, admin: all domains

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
   </DomainImpl>
</data>
```

The response of the atomic action is

```
<data>
   <Domains>
      <DomainImpl>
         <m_DomainName>...</m_DomainName>
         <m_OID>...</m_OID>
      </DomainImpl>
      <DomainImpl>
         <m_DomainName>...</m_DomainName>
         <m_OID>...</m_OID>
      </DomainImpl>
      ...
   </Domains>
</data>
```

3.6.37. Service ShowDomainDetailedData

3.6.37.1. Functionality

The ShowDomainDetailedData service executes the `ShowDomainDetailedData` and the `ShowGatewayLocationsForDomain` atomic action.

3.6.37.2. Atomic action ShowDomainDetailedData

The ShowDomainDetailedData atomic action shows queries the domain data for a domain. If the domain to query does not exist, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
```

```
      <m_OID>003FD06DFA-34B2835E-566A6D15</m_OID>
   </DomainImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
      <m_OID>...</m_OID>
      <m_Description>...</m_Description>
   </DomainImpl>
   <Attributes>
         <AttributeImpl m_OID="..." m_Key="..." m_Value="..." />
         <AttributeImpl m_OID="..." m_Key="..." m_Value="..." />
         ...
   </Attributes>
</data>
```

3.6.38. Service ModifyDomainDetailedData

3.6.38.1. Functionality

The ModifyDomainDetailedData service checks if all attributes that were passed with the request belong to the domain. If one or more attributes do not belong to the domain, then a LogicalException with error code DATA_FAULT is thrown. Otherwise, the `ModifyDomainDetailedData` atomic action is executed.

3.6.38.2. Atomic action ModifyDomainDetailedData

The ModifyDomainDetailedData atomic action calls checkMaxRegisteredUser of the derived abstract class (throws exception in case the domain must not be created) and updates the one record of the domainimpl database table and zero or more records of the attributeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
      <m_Description>...</m_Description>
   </DomainImpl>
   <Attributes>
      <AttributeImpl>
         <m_OID>...</m_OID>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      <AttributeImpl>
         <m_OID>...</m_OID>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      ...
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.39. Service ShowRoutingRulesForDomain

3.6.39.1. Functionality

The ShowRoutingRulesForDomain executes the ShowRoutingRulesForDomain atomic action.

3.6.39.2. Atomic action ShowRoutingRulesForDomain

The ShowRoutingRulesForDomain atomic action queries the routing rules for a domain from the database. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <RoutingRules>
      <RoutingRulImpl>
         <m_OID>...</m_OID>
         <m_Type>RoutingRule</m_Type>
         <m_ParentOID>...</m_ParentOID>
         <m_Order>...</m_Order>
         <m_Action>...</m_Action>
         <m_PatternFrom>...</m_PatternFrom>
         <m_PatternTo>...</m_PatternTo>
         <m_PatternRequestUri>
            <![CDATA[...]]></m_PatternRequestUri>
         <m_Attribute>...</m_Attribute>
         <m_RequeryResult>...</m_RequeryResult>
         <m_FallThroughOnFail>...</m_FallThroughOnFail>
         <m_NightMode>...</m_NightMode>
         <m_TerminatingRule>...</m_TerminatingRule>
      </RoutingRulImpl>
      <RoutingRulImpl>
         <m OID>...</m OID>
         <m_Type>RoutingRule</m_Type>
         <m_ParentOID>...</m_ParentOID>
         <m_Order>...</m_Order>
         <m_Action>...</m_Action>
         <m_PatternFrom>...</m_PatternFrom>
         <m_PatternTo>...</m_PatternTo>
         <m_PatternRequestUri>
            <![CDATA[...]]></m_PatternRequestUri>
         <m_Attribute>...</m_Attribute>
         <m_RequeryResult>...</m_RequeryResult>
         <m_FallThroughOnFail>...</m_FallThroughOnFail>
         <m_NightMode>...</m_NightMode>
         <m_TerminatingRule>...</m_TerminatingRule>
      </RoutingRulImpl>
      ...
   </RoutingRules>
</data>
```

3.6.40. Service ChangeRoutingRuleForDomain

3.6.40.1. Functionality

The ChangeRoutingRuleForDomain executes the ChangeRoutingRuleForDomain atomic action.

3.6.40.2. Atomic action ChangeRoutingRuleForDomain

The ChangeRoutingRuleForDomain atomic action first checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Then the atomic action checks the order of the routing rule and makes sure that the order of all other routing rules is changed to keep the orders correct. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If the order number is too high order too low, then a LogicalException with error code DATA_FAULT is thrown. Then the record in the routingruleimpl database table is updated. The input if the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>003FD06DFA-34B2835E-566A6D15</m_OID>
   </DomainImpl>
   <RoutingRuleImpl>
      <m_OID>...</m_OID>
      <m_Type>...</m_Type>
      <m_Order>...</m_Order>
      <m_Action>...</m_Action>
      <m_PatternFrom>...</m_PatternFrom>
      <m_PatternTo>...</m_PatternTo>
      <m_PatternRequestUri>
         <![CDATA[...]]>
      </m_PatternRequestUri>
      <m_Attribute>...</m_Attribute>
      <m_RequeryResult>...</m_RequeryResult>
      <m_FallThroughOnFail>...</m_FallThroughOnFail>
      <m_NightMode>...</m_NightMode>
   </RoutingRuleImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.41. Service DeleteRoutingRulesForDomain

3.6.41.1. Functionality.

The DeleteRoutingRulesForDomain service executes the DeleteRoutingRulesForDomain atomic action.

3.6.41.2. Atomic action DeleteRoutingRulesForDomain

The DeleteRoutingRulesForDomain atomic action first checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Then the atomic action checks the order of the routing rule and makes sure that the order of all other routing rules is changed to keep the orders correct. If the order number is too high order too low, then a LogicalException with error code DATA_FAULT is thrown. Then the corresponding records from the routingruleimpl database table are deleted. The input if the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>a95f38f5-10fa-2152-dff7-03896676d6d2</m_OID>
   </DomainImpl>
   <RoutingRules>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
```

```
      </RoutingRuleImpl>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
      </RoutingRuleImpl>
      ...
   </RoutingRules>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.42. Service ListDomainUsers

3.6.42.1. Functionality

The ListDomainUsers service exeutes the ListDomainUsers atomic action. The returned users are sorted alphabetically (lastname, firstname).

3.6.42.2. Atomic action ListDomainUsers

The ListDomainUsers atomic action first checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Otherwise all users who belong to the domain are queried according to the input:

> m_LastName or m_SIPURI
>
> returns only those users, whose lastname or uri contains the given input as a substring

> m_Role
>
> returns only those users, who belong to the given role

> caller role == ‘customeradmin’
>
> In this case, only users belonging to the same domain as the caller are taken into consideration

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <UserImpl>                          optional
      <m_LastName>...</m_LastName>
      or
      <m_SIPURI>...</m_SIPURI>
      or
      <m_Role>...</m_Role>
   </UserImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
   </DomainImpl>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_SIPURI>...</m_SIPURI>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
```

```
            <m_FirstName>...</m_FirstName>
            <m_LastName>...</m_LastName>
            <m_SIPURI>...</m_SIPURI>
         </UserImpl>
         ...
      </Users>
</data>
```

3.6.43. Service AddFeatureCodesForDomain

3.6.43.1. Functionality

The AddFeaturesCodesForDomain service executes the AddFeatureCodesForDomain atomic action.

3.6.43.2. Atomic action AddFeatureCodesForDomain

The atomic action inserts a record into the featurecodeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <Features>
      <FeatureCodeImpl>
         <m_FeatureCode>...</m_FeatureCode>
         <m_FeatureName>...</m_FeatureName>
      </FeatureCodeImpl>
      <FeatureCodeImpl>
         <m_FeatureCode>...</m_FeatureCode>
         <m_FeatureName>...</m_FeatureName>
      </FeatureCodeImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.44. Service ShowFeatureCodesForDomain

3.6.44.1. Functionality

The ShowFeatureCodesForDomain service executes the ShowFeatureCodesForDomain atomic action.

3.6.44.2. Atomic action ShowFeatureCodesForDomain

The ShowFeatureCodesForDomain atomic action queries the feature codes from the featurecodeimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

The response of the atomic action is

```
<data>
   <Features>
      <FeatureCodeImpl>
         <m_OID>...</m_OID>
         <m_FeatureCode>...</m_FeatureCode>
         <m_FeatureName>...</m_FeatureName>
      </FeatureCodeImpl>
      <FeatureCodeImpl>
         <m_OID>...</m_OID>
         <m_FeatureCode>...</m_FeatureCode>
         <m_FeatureName>...</m_FeatureName>
      </FeatureCodeImpl>
      ...
   </Features>
</data>
```

3.6.45. Service ChangeFeatureCodesForDomain

3.6.45.1. Functionality

The ChangeFeatureCodesForDomain first executes the ShowFeatureCodesForDomain atomic action (see 3.6.44.2) to get the existing feature codes. Then it checks if all features that are passed with the request belong to the domain. If a feature code does not belong to the domain, then a LogicalException with error code DATA_FAULT is thrown. Otherwise, the ChangeFeatureCodesForDomain atomic action is executed.

3.6.45.2. Atomic action ChangeFeatureCodesForDomain

The ChangeFeatureCodesForDomain updates the feature codes in the featurecodeimpl databas table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID></m_OID>
   </DomainImpl>
   <Features>
      <FeatureCodeImpl>
         <m_OID> </m_OID>
         <m_FeatureCode></m_FeatureCode>
         <m_FeatureName> </m_FeatureName>
      </FeatureCodeImpl>
      <FeatureCodeImpl>
         <m_OID> </m_OID>
         <m_FeatureCode></m_FeatureCode>
         <m_FeatureName> </m_FeatureName>
      </FeatureCodeImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.46. Service CreateLocalDomainGroup

3.6.46.1. Functionality

The CreateLocalDomainGroup service executes the CreateLocalDomainGroup atomic action.

3.6.46.2. Atomic action CreateLocalDomainGroup

The CreateLocalDomainGroup inserts a record into the groupimpl database table. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <GroupImpl>
      <m_GroupName>...</m_GroupName>
      <m_Description>...</m_Description>
   </GroupImpl>
</data>
```

The response of the atomic action is

```
<data>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
</data>
```

3.6.47. Service DeleteLocalDomainGroup

3.6.47.1. Functionality

The DeleteLocalDomainGroup service checks if the group type of the group to delete is `Constants.GROUP_TYPE_LOCAL`. If the type is not `Constants.GROUP_TYPE_LOCAL`, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the DeleteLocalDomainGroup atomic action is executed.

3.6.47.2. Atomic action DeleteLocalDomainGroup

The DeleteLocalDomainGroup deletes a record from the groupimpl database table using the DeleteTables atomic action(see 3.4). The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.48. Service AddUsersToLocalDomainGroup

3.6.48.1. Functionality

The AddUsersToLocalDomainGroup service uses the getVerificationData method (see 3.5.5.1) to get the group type. If the group type not is `Constants.GROUP_TYPE_LOCAL`, then a LogicalException with error code DATA_FAULT is thrown, otherwise the AddUsersToLocalDomainGroup atomic action is executed.

3.6.48.2. Atomic action AddUsersToLocalDomainGroup

The AddUsersToLocalDomainGroup checks if the domain exists. If the domain does not exist then a LogicalException with a DATA_FAULT exception is thrown. Otherwise the atomic action checks if all users defined in the request are in the domain. If this is not the case, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the database is updated. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
      </UserImpl>
   </Users>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.49. Service DeleteUsersFromLocalDomainGroup

3.6.49.1. Functionality

The DeleteUsersFromLocalDomainGroup executes the DeleteUsersFromLocalDomainGroup atomic action.

3.6.49.2. Atomic action DeleteUsersFromLocalDomainGroup

The DeleteUsersFromLocalDomainGroup checks if a domain for the group exists in the database. If no domain can be found, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the atomic action checks if all users defined in the request are in the domain. If this is not the case, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the database entries are deleted. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
      </UserImpl>
   </Users>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.50. Service ListUsersOfLocalDomainGroup

3.6.50.1. Functionality

The ListUserOfLocalDomainGroup service exeutes the ListUserOfLocalDomainGroup . It lists all users of a input local domain group. The returned users are sorted alphabetically (lastname, firstname).

3.6.50.2. Atomic action ListDomainUsers

The ListUserOfLocalDomainGroup atomic action first checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Otherwise all users who belong to the input local domain group are queried.

The input of the atomic action is

```
<data>
 <Caller uri="admin@kapsch.net" session="1" />
 <GroupImpl>
    <m_OID>003FD08B4E-F3BE690B-G2</m_OID>
 </GroupImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
   </DomainImpl>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_SIPURI>...</m_SIPURI>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_SIPURI>...</m_SIPURI>
      </UserImpl>
      ...
   </Users>
</data>
```

3.6.51. Service ListLocalDomainGroups

3.6.51.1. Functionality

The ListLocalDomainGroups service executes the ListLocalDomainGroups atomic action.

3.6.51.2. Atomic action ListLocalDomainGroups

The ListLocalDomainGroups checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Otherwise it queries the groups from the database. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

The output of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
   </DomainImpl>
   <Groups>
      <GroupImpl>
         <m_OID>...</m_OID>
         <m_GroupName>...</m_GroupName>
         <m_Description>...</m_Description>
         <m_GroupType>...</m_GroupType>
      </GroupImpl>
      <GroupImpl>
         <m_OID>...</m_OID>
         <m_GroupName>...</m_GroupName>
         <m_Description>...</m_Description>
         <m_GroupType>...</m_GroupType>
      </GroupImpl>
      ...
   </Groups>
</data>
```

3.6.52. Service CreateDomainGroup

3.6.52.1. Functionality

The CreateDomainGroup service executes the CreateDomainGroup atomic action.

3.6.52.2. Atomic action CreateDomainGroup

The CreateDomainGroup atomic action inserts the domain group into the groupimpl database table. The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
   <GroupImpl>
      <m_GroupName>...</m_GroupName>
      <m_Description>...</m_Description>
      <m_GroupType>MainRoutingGroup</m_GroupType>
   </GroupImpl>
</data>
```

The response from the atomic action is

```
<data>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
</data>
```

3.6.53. Service DeleteDomainGroup

3.6.53.1. Functionality

The DeleteDomainGroup service checks if the group type of the group to delete is `Constants.GROUP_TYPE_DOMAIN_FEATURES` or `Constants.GROUP_TYPE_DOMAIN_ROUTING`. Otherwise a LogicalException with error code DATA_FAULT is thrown. Otherwise the DeleteDomainGroup atomic action is executed.

3.6.53.2. Atomic action DeleteDomainGroup

The DeleteDomainGroup atomic action checks if there are still users in this group. If there are users in the group, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the group is deleted from the database using the DeleteTables atomic action (see 3.4). The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.54. Service AddRoutingRuleForDomainGroup

3.6.54.1. Functionality

The AddRoutingRuleForDomainGroup service executes the AddRoutingRuleForDomainGroup atomic action.

3.6.54.2. Atomic action AddRoutingRuleForDomainGroup

The AddRoutingRuleForDomainGroup atomic action inserts one record into the routingruleimpl database table. The atomic action maintains the order as defined in the database. For example: if two records exist in the database with order=1 and order=2, and a new record is to be inserted at position 2, then the record that was originally at order=2 will move to order=3. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If a too high or too low order is specified, then a LogicalException with error code DATA_FAULT is thrown. Also, if the domain can not be found in the database, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <RoutingRuleImpl>
      <m_Type>RoutingRule</m_Type>
      <m_Order>...</m_Order>
      <m_Action>...</m_Action>
      <m_PatternFrom>...</m_PatternFrom>
      <m_PatternTo>....</m_PatternTo>
      <m_PatternRequestUri>
         <![CDATA[...]]>
      </m_PatternRequestUri>
      <m_Attribute>...</m_Attribute>
      <m_RequeryResult>...</m_RequeryResult>
      <m_FallThroughOnFail>...</m_FallThroughOnFail>
      <m_NightMode>...</m_NightMode>
      <m_TerminatingRule>...</m_TerminatingRule>
   </RoutingRuleImpl>
</data>
```

The response of the atomic action is:

```
<data>
   <RoutingRuleImpl>
      <m_OID>...</m_OID>
   </RoutingRuleImpl>
</data>
```

3.6.55. Service ShowRoutingRulesForDomainGroup

3.6.55.1. Functionality

The ShowRoutingRulesForDomainGroup service executes the ShowRoutingRulesForDomainGroup atomic action.

3.6.55.2. Atomic action ShowRoutingRulesForDomainGroup

The ShowRoutingRulesForDomainGroup atomic action queries the routing rules for a domain group from the database. If the group does not exist, then a LogicalException with error code DATA_FAULT is thrown. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
      <m_OID>...</m_OID>
   </DomainImpl>
   <RoutingRules>
      <RoutingRulImpl>
         <m_OID>...</m_OID>
         <m_Type>RoutingRule</m_Type>
         <m_ParentOID>...</m_ParentOID>
         <m_Order>...</m_Order>
         <m_Action>...</m_Action>
         <m PatternFrom>...</m PatternFrom>
         <m_PatternTo>...</m_PatternTo>
         <m_PatternRequestUri>
            <![CDATA[...]]></m_PatternRequestUri>
         <m_Attribute>...</m_Attribute>
         <m_RequeryResult>...</m_RequeryResult>
         <m_FallThroughOnFail>...</m_FallThroughOnFail>
         <m_NightMode>...</m_NightMode>
         <m_TerminatingRule>...</m_TerminatingRule>
      </RoutingRulImpl>
      <RoutingRulImpl>
         <m_OID>...</m_OID>
         <m_Type>RoutingRule</m_Type>
         <m_ParentOID>...</m_ParentOID>
         <m_Order>...</m_Order>
         <m_Action>...</m_Action>
         <m_PatternFrom>...</m_PatternFrom>
         <m_PatternTo>...</m_PatternTo>
         <m_PatternRequestUri>
            <![CDATA[...]]></m_PatternRequestUri>
         <m_Attribute>...</m_Attribute>
         <m_RequeryResult>...</m_RequeryResult>
         <m_FallThroughOnFail>...</m_FallThroughOnFail>
         <m_NightMode>...</m_NightMode>
         <m_TerminatingRule>...</m_TerminatingRule>
      </RoutingRulImpl>
      ...
```

```
   </RoutingRules>
</data>
```

3.6.56. Service ChangeroutingRuleForDomainGroup

3.6.56.1. Functionality

The ChangeroutingRuleForDomainGroup service executes the ChangeroutingRuleForDomainGroup atomic action.

3.6.56.2. Atomic action ChangeRoutingRuleForDomainGroup

The ChangeRoutingRuleForDomainGroup atomic action first checks if the group exists. If the group does not exist, then a LogicalException with error code DATA_FAULT is thrown. Then the atomic action checks the order of the routing rule and makes sure that the order of all other routing rules is changed to keep the orders correct. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If the order number is too high order too low, then a LogicalException with error code DATA_FAULT is thrown. Then the record in the routingruleimpl database table is updated. The input if the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <RoutingRuleImpl>
      <m_OID>...</m_OID>
      <m_Type>RoutingRule</m_Type>
      <m_Order>...</m_Order>
      <m_Action>...</m_Action>
      <m_PatternFrom>...</m_PatternFrom>
      <m_PatternTo>...</m_PatternTo>
      <m_PatternRequestUri>
         <![CDATA[...]]>
      </m_PatternRequestUri>
      <m_Attribute>...</m_Attribute>
      <m_RequeryResult>...</m_RequeryResult>
      <m_FallThroughOnFail>...</m_FallThroughOnFail>
      <m_NightMode>...</m_NightMode>
   </RoutingRuleImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.57. Service DeleteRoutingRulesForDomainGroup

3.6.57.1. Functionality

The DeleteRoutingRulesForDomainGroup executes the DeleteRoutingRulesForDomainGroup atomic action.

3.6.57.2. Atomic action DeleteRoutingRulesForDomainGroup

The DeleteRoutingRulesForDomainGroup atomic action first checks if the group exists. If the group does not exist, then a LogicalException with error code DATA_FAULT is thrown. Then the atomic action checks the order of the routing rule and makes sure that the order of all other routing

rules is changed to keep the orders correct. To prevent parallel atomic actions, all SELECTS are FOR UPDATE. If the order number is too high order too low, then a LogicalException with error code DATA_FAULT is thrown. Then the corresponding records from the routingruleimpl database table are deleted. The input if the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>a95f38f5-10fa-2152-dff7-03896676d6d2</m_OID>
   </GroupImpl>
   <RoutingRules>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
      </RoutingRuleImpl>
      <RoutingRuleImpl>
         <m_OID>...</m_OID>
      </RoutingRuleImpl>
      ...
   </RoutingRules>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.58. Service ListDomainGroups

3.6.58.1. Functionality

The ListDomainGroups service executes the ListDomainGroups atomic action.

3.6.58.2. Atomic action ListDomainGroups

The ListDomainGroups checks if the domain exists. If the domain does not exist, then a LogicalException with error code DATA_FAULT is thrown. Otherwise it queries the groups from the database. The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <DomainImpl>
      <m_OID>...</m_OID>
   </DomainImpl>
</data>
```

the response from the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>..</m_DomainName>
   </DomainImpl>
   <Groups>
      <GroupImpl
         <m_OID>...</m_OID>
         <m_GroupName>...</m_GroupName>
         <m_Description>...</m_Description>
         <m_GroupType>...</m_GroupType>
      </GroupImpl
   </Groups>
</data>
```

3.6.59. Service AddFeaturesForDomainGroup

3.6.59.1. Functionality

The AddFeaturesForDomainGroup service executes the AddFeaturesForDomainGroup atomic action.

3.6.59.2. Atomic action AddFeaturesForDomainGroup

The AddFeaturesForDomainGroup atomic action inserts the features and the corresponding attributes into the database. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>OID</m_OID>
   </GroupImpl>
   <Features>
      <FeatureImpl>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_Key="..." m_Value="..." />
         <Attribute m_Key="..." m_Value="..." />
      </FeatureImpl>
      <FeatureImpl>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attribute m_Key="..." m_Value="..." />
         <Attribute m_Key="..." m_Value="..." />
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.60. Service ShowFeaturesForDomainGroup

3.6.60.1. Functionality

The ShowFeaturesForDomainGroup service executes the ShowFeaturesForDomainGroup atomic action.

3.6.60.2. Atomic action ShowFeaturesForDomainGroup

The ShowFeaturesForDomainGroup queries the features from the database. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>ae48b0b3-0bfa-b0ea-dfbd-73a988043f9c</m_OID>
   </GroupImpl>
</data>
```

The response of the atomic action is

```
<data>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
```

```
            <m_Type>...</m_Type>
            <m_FeatureName>...</m_FeatureName>
            <m_Status>...</m_Status>
            <m_OverridingUserFeature>...<m_OverridingUserFeature>
            <Attribute m_OID="..." m_Type="..." m_Key="..." m_Value="..." />
            <Attribute m_OID="..." m_Type="..." m_Key="..." m_Value="..." />
            ...
         </FeatureImpl>
         <FeatureImpl>
            <m_OID>...</m_OID>
            <m_Type>...</m_Type>
            <m_FeatureName>...</m_FeatureName>
            <m_Status>...</m_Status>
            <m_OverridingUserFeature>...<m_OverridingUserFeature>
            <Attribute m_OID="..." m_Type="..." m_Key="..." m_Value="..." />
            <Attribute m_OID="..." m_Type="..." m_Key="..." m_Value="..." />
            ...
         </FeatureImpl>
         ...
      </Features>
</data>
```

3.6.61. Service ModifyFeaturesForDomainGroup

3.6.61.1. Functionality

The ModifyFeaturesForDomainGroup service checks if all features and attributes exist and belong to the domain group. If not, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the ModifyFeaturesForDomainGroup atomic action is executed.

3.6.61.2. Atomic action ModifyFeaturesForDomainGroup

The atomic action updates the features and adds/modifies/deletes the attributes. The input for the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_FeatureName>...</m_FeatureName>
         <m_Status>...</m_Status>
         <m_OverridingUserFeature>...</m_OverridingUserFeature>
         <Attributes>
            <Add>
               <Attribute m_Key="..." m_Value="..." />
            </Add>
            <Modify>
               <Attribute m_OID="..." m_Key="..." m_Value="..." />
            </Modify>
            <Delete>
               <Attribute m_OID="..." />
            </Delete>
         </Attributes>
      </FeatureImpl>
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.62. Service ChangeGroupFeatureStatus

3.6.62.1. Functionality

The ChangeGroupFeatureStatus service checks if all features belong to the group. If not, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the ChangeGroupFeatureStatus atomic action is executed.

3.6.62.2. Atomic action ChangeGroupFeatureStatus

The ChangeGroupFeatureStatus atomic action updates the features in the database. The input if the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Status>...</m_Status>
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
         <m_Status>...</m_Status>
      </FeatureImpl>
      ...
   </Features>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.63. Service DeleteFeaturesForDomainGroup

3.6.63.1. Functionality

This services deletes all given features of the given FEATURE DOMAIN Group. The service

- uses the `getVerification` method (see 3.5.5.1) to get the GroupType value to check if the GroupImpl value of is `Constants.GROUP_TYPE_DOMAIN_FEATURE`. If not, then a LogicalException with error code DATA_FAULT is thrown.
- calls a.a. `ShowFeaturesForDomainGroup` to get all features for the given domain. It checks, whether the features in the input XML belong to the user (compares both lists) and throws a LogicalException with error code LOGIC_ERROR if not
- calls a.a. `DeleteFeaturesForDomainGroup` to delete the desired features/attributes in tabel `featureimpl/attributeimpl`
- calls a.a. `ShowFeaturesForDomainGroup` to get all remaining features for the given domain in order to return it to the caller.

3.6.63.2. Atomic action DeleteFeaturesForDomainGroup

The `DeleteFeaturesForDomainGroup` atomic action deletes all features/attributes belonging to the domain.

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Features>
      <FeatureImpl>
         <m_OID>...</m_OID>
      </FeatureImpl>
      <FeatureImpl>
         <m_OID>...</m_OID>
      </FeatureImpl>
      ...
   </Features>
</data>
```

3.6.64. Service ListUsersOfDomainGroup

3.6.64.1. Functionality

The ListUsersOfDomainGroup service uses the getVerification method (see 3.5.5.1) to get the GroupType value to check if the GroupImpl value of is `Constants.GROUP_TYPE_DOMAIN_ROUTNG`. If not, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the ListUsersOfDomainGroup atomic action is executed.

3.6.64.2. Atomic action ListUsersOfDomainGroup

The ListUsersOfDomainGroup atomic action checks if the domain exists. If not, then a LogicalException with error code DATA_FAULT is thrown. Otherwise the users are queried from the database. A (part of) last name can be specified to search for a user. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
   </GroupImpl>
   <UserImpl>
      <m_LastName>...</m_LastName>
   </UserImpl>
</data>
```

The response of the atomic action is

```
<data>
   <DomainImpl>
      <m_DomainName>...</m_DomainName>
   </DomainImpl>
   <GroupImpl>
      <m_OID>...</m_OID>
      <m_GroupName>..</m_GroupName>
      <m_GroupType>...</m_GroupType>
   </GroupImpl>
   <Users>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_SIPURI>...</m_ SIPURI>
      </UserImpl>
      <UserImpl>
         <m_OID>...</m_OID>
         <m_FirstName>...</m_FirstName>
         <m_LastName>...</m_LastName>
         <m_SIPURI>...</m_ SIPURI>
      </UserImpl>
```

```
      ...
   </Users>
</data>
```

3.6.65. ServiceExecutePhoneService

3.6.65.1. Functionality

This service is called by all Phone services, which execute a perls script on a UNIX host. The name of the calling service is tunneld within the source field of the TICE request. The location of the script is taken from the configuration variable `PhoneServiceScriptLocationPrefix`, defined in `.../sip/etc/ SIP_Conf.xml` of the installation.

`ExecutePhoneService` analyzes the input request and

> calls appropriate atomic action:
   1. `CheckPhoneForUser` – checks, whether user of the input exists in SIP DB, (throws DATAException if not). Is called for service `Create/DeletePhoneForUser`
   2. `CheckPhoneForDomain` – checks, whether domain of the input exists in SIP DB, (throws DATAException if not). Is called for service `Recreate/DeletePhoneForDomain`
   3. `CheckPhoneForLocation` – checks, whether location of the input exists in SIP DB, (throws DATAException if not). Is called for service `Recreate/DeletePhoneForLocation`
   4. `CheckAddressBook` – checks, whether user of the input exists in SIP DB, (throws DATAException if not). Is called for service `Create/DeleteAddressBook`
   5. `CheckIADForUsers` – checks, whether user of the input exists in SIP DB (must be all in the same local group of input), (throws DATAException if not). Is called for service `Create/DeleteIADForUsers`

> generates Perl script invocation command (the parameters of the script invocation are put into quotes for secutrity reasons).
   6. the name of the perl script is taken from the service name
   7. in case of Service `Create/DeleteIADForUsers` the parameters `m_SIPURI` of the list are numbered for the perl script call (`m_SIPURI1, m_SIPURI2, ...`)

> calls atomic action `kapsch.resource.ssh.atomicaction.Execute` which opens a SecureSHell (SSH) on the addressed HOST and exceutes the script of the comand string.

3.6.65.2. Atomic Action CheckPhoneForUser

This a.a. checks, whether the given user exists in the SIP DB and throws a `LOGICAL` Execption if not. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Phone_Attr>
      <m_SIPURI>...</m_SIPURI>
   </Phone_Attr>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.65.3. Atomic Action CheckPhoneForDomain

This a.a. checks, whether the given domain exists in the SIP DB and throws a `LOGICAL` Execption if not. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Domain_Attr>
      <domain>...</domain>
   </Domain_Attr >
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.65.4. Atomic Action CheckPhoneForUser

This a.a. checks, whether the given location exists in the SIP DB and throws a `LOGICAL` Execption if not. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Location_Attr>
      <location>...</location>
   </Location _Attr >
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.65.5. Atomic Action CheckAddressBook

This a.a. checks, whether the given user exists in the SIP DB and throws a `LOGICAL` Execption if not. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <AddressBook>
      <m_SIPURI>...</m_SIPURI>
   </AddressBook>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.65.6. Atomic Action CheckIADForUsers

This a.a. checks, whether the given users all belong to the given local group. The configurartion parameter `MaxIADUsers` defines, how many users are allowed in the group. The users found in the DB must match the users given by the input and both sets must not exceed the `MaxUIADUsers` number.

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
<IADForUsers>
         <MAC_Addr>4711</MAC_Addr>
         <m_SIPURI>1111@kcc.at</m_SIPURI>
         <m_SIPURI>123@kcc.at</m_SIPURI>
         <m_SIPURI>2222@kcc.at</m_SIPURI>
         <m_SIPURI>234@kcc.at</m_SIPURI>
         <IAD_Type>MP102-FXs</IAD_Type>
```

```
        <m_OID>754eb32a-e7ff-1ba1-eab7-565d6b101e07</m_OID>
      </IADForUsers>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.66. Service CreatePhoneForUser

3.6.66.1. Functionality

The CreatePhoneForUser service executes the ExecutePhoneService service (see 3.6.65).

3.6.67. Service DeletePhoneForUser

3.6.67.1. Functionality

The DeletePhoneForUser service executes the ExecutePhoneService service (see 3.6.65).

3.6.68. Service RecreatePhoneForDomain

3.6.68.1. Functionality

The RecreatePhoneForLocation service executes the ExecutePhoneService service (see 3.6.65).

3.6.69. Service DeletePhoneForDomain

3.6.69.1. Functionality

The DeletePhoneForDomain service executes the ExecutePhoneService service (see 3.6.65).

3.6.70. Service RecreatePhoneForLocation

3.6.70.1. Functionality

The RecreatePhoneForLocation service executes the ExecutePhoneService service (see 3.6.65).

3.6.71. Service DeletePhoneForLocation

3.6.71.1. Functionality

The DeletePhoneForLocation service executes the ExecutePhoneService service (see 3.6.65).

3.6.72. Service CreateAddressBook

3.6.72.1. Functionality

The CreateAddressBook service executes the ExecutePhoneService service (see 3.6.65).

3.6.73. Service DeleteAddressBook

3.6.73.1. Functionality

The DeleteAddressBook service executes the ExecutePhoneService service (see 3.6.65).

3.6.74. Service SetDomainPhoneSettings

3.6.74.1. Functionality

This service

- > call atomic action `CheckPhoneForDomain`, in order to check whether the given domain exists in the SIP DB
- > calls atomic action `SetDomainPhoneSettings`.

3.6.74.2. Atomic action SetDomainPhoneSettings

The atomic action `SetDomainPhoneSettings` inserts the phone settings into the domain_attr table of the Phone DB. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Domain_Attr>
      <domain>...</domain>
   </Domain_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.75. Service DeleteDomainPhoneSettings

3.6.75.1. Functionality

This service

- > call atomic action `CheckPhoneForDomain`, in order to check whether the given domain exists in the SIP DB
- > calls atomic action `DeleteDomainPhoneSettings`.

3.6.75.2. Atomic action DeleteDomainPhoneSettings

The atomic action `DeleteDomainPhoneSettings` deletes records from the domain_attr table of the Phone DB. The input of the atomic action is

```
<data>
```

```
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Domain_Attr>
      <domain>customer.at</domain>
   </Domain_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.76. Service ShowDomainPhoneSettings

3.6.76.1. Functionality

This service shows all entries of table domain_attr for a given domain. It calls atomic action `ShowDomainPhoneSettings`.

3.6.76.2. Atomic action ShowDomainPhoneSettings

The atomic action `ShowDomainPhoneSettings` selects all records of table domain_attr in the Phone DB for a given domain. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Domain_Attr>
      <domain>customer.at</domain>
   </Domain_Attr>
</data>
```

The response of the atomic action is:

```
<data>
   <Domain_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Domain_Attr>
<data>
```

3.6.77. Service SetLocationPhoneSettings

3.6.77.1. Functionality

This service

- > call atomic action `CheckPhoneForLocation`, in order to check whether the given location exists in the SIP DB
- > calls atomic action `SetLocationPhoneSettings`.

3.6.77.2. Atomic action SetLocationPhoneSettings

The atomic action `SetLocationPhoneSettings` inserts the location settings into the location_attr table of the Phone DB. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   </Location_Attr>
      <location>...</location>
   </Location_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
      ...
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.78. Service DeleteLocationPhoneSettings

3.6.78.1. Functionality

This service

- call atomic action `CheckPhoneForLocation`, in order to check whether the given location exists in the SIP DB
- calls atomic action `DeleteLocationPhoneSettings`.

3.6.78.2. Atomic action DeleteLocationPhoneSettings

The atomic action `DeleteDomainPhoneSettings` deletes records from the `location_attr` table of the Phone DB. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   </Location_Attr>
      <location>...</location>
   </Location_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.79. Service ShowLocationPhoneSettings

3.6.79.1. Functionality

This service shows all entries of table location_attr for a given location. It calls atomic action `ShowLocationPhoneSettings`.

3.6.79.2. Atomic action ShowLocationPhoneSettings

The atomic action `ShowLocationPhoneSettings` selects all records of table location_attr in the Phone DB for a given location. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Location_Attr>
      <location>...</location>
   </ Location _Attr>
</data>
```

The response of the atomic action is:

```
<data>
   <Location_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Location_Attr>
<data>
```

3.6.80. Service SetUserPhoneSettings

3.6.80.1. Functionality

This service

- > call atomic action `CheckPhoneForUser`, in order to check whether the given location exists in the SIP DB
- > calls atomic action `SetLocationPhoneSettings`.

3.6.80.2. Atomic action SetUserPhoneSettings

The atomic action `SetLocationPhoneSettings` inserts the location settings into the location_attr table of the Phone DB. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   </Location_Attr>
      <location>...</location>
   </Location_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
         <Attr_Value>...</Attr_Value>
      </Attribute>
      ...
   </Attributes>
```

```
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.81. Service DeleteUserPhoneSettings

3.6.81.1. Functionality

This service

- > call atomic action `CheckPhoneForLocation`, in order to check whether the given location exists in the SIP DB
- > calls atomic action `DeleteUserPhoneSettings`.

3.6.81.2. Atomic action DeleteUserPhoneSettings

The atomic action `DeleteUserPhoneSettings` deletes records from the `location_attr` table of the Phone DB. The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   </Location_Attr>
      <location>...</location>
   </Location_Attr>
   <Attributes>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Attributes>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.82. Service ShowUserPhoneSettings

3.6.82.1. Functionality

This service shows all entries of table phone_attr for a given user (MAC_Address). It calls atomic action `ShowUserPhoneSettings`.

3.6.82.2. Atomic action ShowUserPhoneSettings

The atomic action `ShowUserPhoneSettings`

- > selects all records of table `phone_attr` in the Phone DB for a given user (MAC_Addr) and all records of table `user_attr` (where m_SIPURI is taken from the belonging phone_attr entry).
- > calls a.a. `ShowLocationPhoneSettings`
- > calls a.a. `ShowDomainPhoneSettings`

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
```

```
   </LocationImpl>
   <Phone_Attr>
      <MAC_Addr >...</MAC_Addr>
   </Phone_Attr>
</data>
```

The response of the atomic action is:

```
<data>
   <Phone_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Phone_Attr>
   <User_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </User_Attr>
   <Domain_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Domain_Attr>
   <Location_Attr>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
      <Attribute>
         <Attr_Name>...</Attr_Name>
      </Attribute>
   </Location_Attr>
<data>
```

3.6.83. Service ChangePhoneSettingsKey

3.6.83.1. Functionality

This generic service changes the content of the primary key column in table location_attr (location), domain_attr (domain) and user_attr (m_SIPURI). The service is implemented by the generic service GenericDBService. It calls atomic action `ChangePhoneSettingsKey`.

3.6.83.2. Atomic action ChangePhoneSettingsKey

The atomic action `ChangePhoneSettingsKey`

> according to the input (`LOCATION|DOMAIN|USER`) it updates the primary key field of the corresponding table (`location_attr.location, domain_attr.domain, user_attr.m_SIPURI`).

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <Key>
```

```
      <Table>LOCATION|DOMAIN|USER</Table>
      <OldValue>…</OldValue>
      <NewValue>…</NewValue>
   </Key>
</data>
```

The response of the atomic action is:

```
</data>
```

3.6.84. Service ShowbillingRecords

3.6.84.1. Functionality

This service shows all billing records for a m_SIPURI.

TODO

3.6.84.2. Atomic action ShowBillingRecords

TODO

3.6.85. Service AddProviderLocation

3.6.85.1. Functionality

This service creates a new provider location using the AddProviderLocation atomic action.

3.6.85.2. Atomic action AddProviderLocation

This atomic action inserts the provider location into the database. This input of the atomic action is:

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <LocationImpl>
     <m_Location>...</m_ Location >
   </LocationImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.86. Service ChangeProviderLocation

3.6.86.1. Functionality

This service changes an existing provider location using the ChangeProviderLocation atomic action.

3.6.86.2. Atomic action ChangeProviderLocation

This atomic action modifies the provider location in the database. This input of the atomic action is:

```
<data>
```

```
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <LocationImpl>
     <m_OID>...</m_OID>
     <m_Location>...</m_Location>
   </LocationImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.87. Service DeleteProviderLocation (currently disabled, because not designed in FAD)

3.6.87.1. Functionality

This service deletes a provider location from the SIP DB. It calls the atomic action `DeleteProviderLocation`.

3.6.87.2. Atomic action DeleteProviderLocation

This atomic action deletes the provider location inkl. attributes from the database. This input of the atomic action is:

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <LocationImpl>
     <m_OID></m_OID>
   </LocationImpl>
</data>
```

The response of the atomic action is an empty data node: `<data />`.

3.6.88. Service ShowProviderLocations

3.6.88.1. Functionality

This service shows all provider locations in the SIP DB. It calls atomic action `ShowProviderLocation`.

3.6.88.2. Atomic action ShowProviderLocations

This atomic actions selects all entries of table locationimpl in the SIP DB and returns them.

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
</data>
```

The response of the atomic action is:

```
<data>
   <Locations>
     <LocationImpl>
        <m_OID>...</m_OID>
        <m_Location>...</m_ Location>
     </LocationImpl>
     <LocationImpl>
```

```
        <m_OID>...</m_OID>
        <m_Location>...</m_ Location>
      </LocationImpl>
      ...
  </Locations>
<data>
```

3.6.89. Service ShowLocationAttributes

3.6.89.1. Functionality

This service shows all attributes that belong to a location using the ShowLocationAttributes atomic action.

3.6.89.2. Atomic action ShowLocationAttributes

This atomic action queries all location attributes from the SIP database. The input of the atomic action is:

```
<data>
   <LocationImpl>
     <m_OID>...</m_OID>
   </LocationImpl>
<data>
```

The response of the atomic action is:

```
<data>
  <LocationImpl>
    <m_OID>...</m_OID>
  </LocationImpl>
  <Attributes>
    <AttributeImpl>
      <m_Key>...</m_Key>
      <m_Value>...</m_Value>
    </AttributeImpl>
    <AttributeImpl>
      <m_Key>...</m_Key>
      <m_Value>...</m_Value>
    </AttributeImpl>
    ...
  </Attributes
<data>
```

3.6.90. Service AddLocationAttributes

3.6.90.1. Functionality

This service adds attributes to a provider location for the sip proxy local database. It calls atomic action `AddLocationAttributes`.

3.6.90.2. Atomic action AddLocationAttributes

This atomic action loops over all attributes and inserts them in the SIP DB. The data section of the input is:

```
<data>
...
   <LocationImpl>
      <m_OID>...</m_OID>
   </LocationImpl>
   </Attributes>
      <AttributeImpl>
```

```
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      <AttributeImpl>
         <m_Key>...</m_Key>
         <m_Value>...</m_Value>
      </AttributeImpl>
      ...
   </Attributes>
</data>
```

The data section of the response of the atomic action is empty.

3.6.91. Service ChangeLocationAttribute

3.6.91.1. Functionality

This service modifies one attribute of a provider location for the sip proxy local database. It calls atomic action `ChangeLocationAttribute`.

3.6.91.2. Atomic action ChangeLocationAttribute

This atomic action modifies one attribute in the SIP DB. The data section of the input is:

```
<data>
...
   <LocationImpl>
      <m_OID>...</m_OID>
   </LocationImpl>
   <AttributeImpl>
      <m_OID>...</m_OID>
      <m_Key>...</m_Key>
      <m_Value>...</m_Value>
   </AttributeImpl>
</data>
```

The data section of the response of the atomic action is empty.

3.6.92. Service DeleteLocationAttribute

3.6.92.1. Functionality

This service deletes one attribute of a provider location for the sip proxy local database. It calls atomic action `ChangeLocationAttribute`.

3.6.92.2. Atomic action DeleteLocationAttribute

This atomic action deletes one attribute in the SIP DB. The data section of the input is:

```
<data>
...
   <LocationImpl>
      <m_OID>...</m_OID>
   </LocationImpl>
   <AttributeImpl>
      <m_OID>...</m_OID>
   </AttributeImpl>
</data>
```

The data section of the response of the atomic action is empty.

3.6.93. Service ProxyServerCommand

3.6.93.1. Functionality

This service executes the script sipproxy.sh on a remote HOST. The service

> calls atomic action ProxyServerLocate in order to locate the remote HOST
> generates the script invocation call
  8. Location of the script can be configured in the configuration file SIP_Config.xml.
  9. the input parameter for the script invocation is read out of the TICE request and added to the call
> calls atomic action `kapsch.resource.ssh.atomicaction.Execute` which opens a SecureSHell (SSH) on the addressed remote HOST and exceutes the script of the comand string.
  10. Each command which shall be executed in the SSH session is input in a separate line.

3.6.93.2. Atomic action ProxyServerLocate

This atomic action retrieves the OCM name of the remote HOST from the SIP DB. The remote HOST is location dependend, that means, for each location in the SIP DB exist attributes with names BILOXY1|2|3... The value of the attributes are the logical OCM names of the remote HOSTs to be addressed. and are returned in the <ProxyServer> element of the TICE response.

The input of the atomic action is

```
<data>
   <LocationImpl>
      <Attribute m_Key="JNDI" m_Value="..."/>"
   </LocationImpl>
   <ProxyServer id=”BILOXY1”/>
</data>
```

The response of the atomic action is:

```
<data>
   <ProxyServer id=”BILOXY1” destination=”xxx”/>
</data>
```

3.6.94. Service SshAccessOnSBC

3.6.94.1. Functionality

This service executes a command on the Session Border Controller (SBC) from Jasomi. The service opens an SSH (SecureShell) on the SBC host and issues the command. As the SBC has a restricted subset of available commands (no UNIX commands), a special SBC OCM is used for communication. The service

> prepares the data section
> calls atomic action `kapsch.resource.ssh.atomicaction.SBCExecute` which opens a SecureSHell (SSH) on the addressed remote HOST and exceutes the script of the comand string.
  11. Only one command per call is allowed

> parses the outcome of the command with a search string and returns all lines, where the search string matches as a perl5 pattern.

3.6.95. Service AddAnnouncementsForDomain

3.6.95.1. Functionality

The AddAnnouncementsForDomain service adds announcement entries for a domain in the SIP DB. It executes the `AddAnnouncementsForDomain` atomic action.

3.6.95.2. Atomic action AddAnnouncementsForDomain

The `AddAnnouncementsForDomain` atomic action inserts one record in table announcementsimpl for each announcement in the list. The input of the atomic action is

```
<data>
   <Caller uri="2@gg.at" session="1"/>
      <DomainImpl>
         <m_OID>2501d8ef-aefe-4f06-e7c3-c4fec89c0ce7</m_OID>
      </DomainImpl>
      <AnnouncementList>
         <AnnouncementImpl>
           <m_Type>Announcement</m_Type>
           <m_ContentServer>testcont.sip.net</m_ContentServer>
           <m_WaveFileName>testWaveFileName3</m_WaveFileName>
           <m_Description>blabla</m_Description>
         </AnnouncementImpl>
         <AnnouncementImpl>
           <m_Type>Announcement</m_Type>
           <m_ContentServer>testcont.sip.net</m_ContentServer>
           <m_WaveFileName>testWaveFileName4</m_WaveFileName>
           <m_Description>blabla</m_Description>
         </AnnouncementImpl>
       </AnnouncementList>
    </data>
```

The response of the atomic action is an empty data section:

```
</data>
```

3.6.96. Service DeleteAnnouncementsForDomain

3.6.96.1. Functionality

The DeleteAnnouncementsForDomain service deletes announcement entries for a domain in the SIP DB. It executes the `Delete`Announcements`ForDomain` atomic action.

3.6.96.2. Atomic action DeleteAnnouncementsForDomain

The service first calls a.a. `ShowAnnouncementsUsageForDomain`, which checks whether one of the to be deleted announcements is still in use. If one of the announcements is still in use (non empty return list from a.a.), an exception is thrown and the service aborts.

Otherwise, the a.a. `DeleteAnnouncementsForDoamin` is called with all announcements in the list.

The DeleteAnnouncementsForDomain atomic action deletes one record in table announcementimpl for each announcement in the list. The input of the atomic action is

```
<data>
        <Caller uri="2@gg.at" session="1"/>
        <DomainImpl>
          <m_OID>2501d8ef-aefe-4f06-e7c3-c4fec89c0ce7</m_OID>
        </DomainImpl>
        <AnnouncementList>
         <AnnouncementImpl>
           <m_OID>846c9f53-cbf2-7386-b763-f0c08010c4c5</m_OID>
         </AnnouncementImpl>
         <AnnouncementImpl>
           <m_OID>846c9f57-2be2-c26e-f763-f0c08010c4c5</m_OID>
         </AnnouncementImpl>
        </AnnouncementList>
    </data>
```

The response of the atomic action is a empty data section:

```
</data>
```

3.6.97. Service ShowAnnouncementsForDomain

3.6.97.1. Functionality

The ShowAnnouncementsForDomain service shows all announcement entries for all domains in the SIP DB. It executes the ShowAnnouncementsForDomain atomic action. An optional input filter (domain, type) can be used to restrict the number of rows returned

3.6.97.2. Atomic action ShowAnnouncementsForDomain

The ShowAnnouncementsForDomain atomic action shows all records in table announcementimpl for all domains (optional filter: . The input of the atomic action is

```
 <data>
    <Caller uri="2@gg.at" session="1"/>
   <!-- optional -->
    <DomainImpl>
      <m_OID>2501d8ef-aefe-4f06-e7c3-c4fec89c0ce7</m_OID>
    </DomainImpl>
    <!-- optional -->
    <AnnouncementImpl>
     <m_Type>Announcement</m_Type>
    </AnnouncementImpl>
</data>
```

The response of the atomic action is a data section:

```
 <data>
  <Caller session="1" uri="0815@kcc.at"/>
  <AnnouncementList>
    <AnnouncementImpl>
      <m_OID>....</m_OID>
      <m_ParentOID>....</m_ParentOID>
      <m_Type>....</m_Type>
      <m_ContentServer>....</m_ContentServer>
      <m_WaveFileName>....</m_WaveFileName>
      <m_Description>....</m_Description>
    </AnnouncementImpl>
    <AnnouncementImpl>
      <m_OID>....</m_OID>
      <m_ParentOID>....</m_ParentOID>
      <m_Type>....</m_Type>
      <m_ContentServer>....</m_ContentServer>
```

```
      <m_WaveFileName>....</m_WaveFileName>
      <m_Description>....</m_Description>
      <m_Type>....</m_Type>
    </AnnouncementImpl>
    ...
  </AnnouncementList>
</data>
```

3.6.98. Service ShowAnnouncementsUsageForDomain

3.6.98.1. Functionality

The service `ShowAnnouncementsForDomain` shows the usage of announcements for a domain. The input is:

- `DomainImpl.m_OID` – mandatory
- List of `AnnouncementImpl.m_OID`
  Denotes all announcements, for which the usage shall be evaluated. Usage is expressed in the SIP DB as entries in RoutingRuleImpl and AttributeImpl.

The service calls atomic action `ShowAnnouncementsForDomain`.

3.6.98.2. Atomic action ShowAnnouncementsForDomain

The `ShowAnnouncementsForDomain` atomic action shows all records of tables RoutingRuleImpl, AttributeImpl, which indicate, that a given announcement is still in use. The usage is evaluated in the following way:

- domainimpl-->routingruleimpl
  - Number of found rows is returned in element `<RR_D>`
- domainimpl-->groupimpl-->routingruleimpl
  - Number of found rows is returned in element `<RR_G_D>`
- domainimpl-->groupimpl-->featureimpl-->attributeimpl
  - Number of found rows is returned in element `<A_G_D>`
- domainimpl-->userimpl-->featureimpl-->attributeimpl
  - Number of found rows is returned in element `<A_U_D>`

A mapping between the announcement and the routingruleimpl, attributeimpl entries is done via the following logic:

- routingrulimpl.m_Attribute contains announcementimpl.m_WaveFileName as substring
- attributeimpl.m_value contains announcementimpl.m_WaveFileName as substring

The input of the atomic action is

```
<data>
```

```
    <Caller uri="0815@mississippi.at" session="1"/>
    <DomainImpl>
      <m_OID>. . .</m_OID>
    </DomainImpl>
    <!—optional-->
    <AnnouncementList>
      <AnnouncementImpl>
        <m_OID>. . .</m_OID>
      </AnnouncementImpl>
      <AnnouncementImpl>
        <m_OID>. . .</m_OID>
      </AnnouncementImpl>
    </AnnouncementList>
</data>
```

The response of the atomic action is a data section:

```
  <data>
  <Caller session="1" uri="0815@mississippi.at"/>
  <AnnouncementsUsage>
    <AnnouncementImpl>
      <m_OID>. . .</m_OID>
      <m_WaveFileName>MOH1.wav</m_WaveFileName>
      <A_G_D>1</A_G_D>
      <RR_D>2</RR_D>
      <RR_G_D>1</RR_G_D>
    </AnnouncementImpl>
    <AnnouncementImpl>
      <m_OID>. . .</m_OID>
      <m_WaveFileName>MOH3.wav</m_WaveFileName>
      <A_G_D>1</A_G_D>
    </AnnouncementImpl>
  </AnnouncementsUsage>
</data>
```

3.6.99. Service TriggerSyncServer

3.6.99.1. Functionality

The service `TriggerSyncServer` sends messages to a predefined Socket Server (java.socket) using HOST/Port to address the server:

- A list of messages can be send to server:
    - Each message is send separately (in sequential order), each response is awaited synchronously.
    - All responses are send back in the TICE response – if the are error codes != 200 (success from SyncServer), the response.code will be set to ‘FAILED’.
    - The line delimiter can be configured in the OCM configuration file.

The input is:

- `Caller` – mandatory
- List of `Messages`

The service calls atomic action kapsch.resource.simplesocket.Execute of the Simplesocket OCM

3.6.99.2. Atomic action Execute

The `Execute` atomic action of the SimpleSocket OCM:

- converts the TiceRequest into the 'protocol' of the SysncSever
- sends the message via an socket JCA OCM
- converts back the response in a TICE data
- only works for String based messages.
-

The input of the atomic action is

```
<data>
 <Caller uri="0815@mississippi.at" session="1"/>
 <MessageList>
   <Message>
     <Request>MESSAGE;123;kcc.at;CFW</Request>
   </Message>
   <Message>
     <Request>NOTIFY;123;kcc.at;REBOOT</Request>
   </Message>
 </MessageList>
</data>
```

The response of the atomic action is a data section:

```
<data>
 <Caller uri="0815@mississippi.at" session="1"/>
 <MessageList>
   <Message>
     <Request>MESSAGE;123;kcc.at;CFW</Request>
     <Response>200</Response>
   </Message>
   <Message>
     <Request>NOTIFY;123;kcc.at;REBOOT</Request>
     <Response>999</Response>
   </Message>
 </MessageList>
</data>
```

3.6.100. Service SendEMail

3.6.100.1. Functionality

The service `SendEMail` sends eMail messages to a list of input recipients via the NTX internal SMTP logger (uses log4J SMTP appender). The sender eMail address is selected from the caller in the SIP DB. Subject and Textmessage are wrapped in a CDATA section to allow for special characters. The eMail is send asynchronous.

The input is:

```
<data>
```

```
  <Caller uri="admin@defaultdomain" session="1"/>
  <eMail>
    <ToList>
     <To>max.mustermann@gmx.at</To>
     <To>…</To>
    </ToList>
 <Subject><![CDATA[Subject Service SendEMail]]></Subject>
 <Message><![CDATA[Message]]></Message>
  </eMail>
</data>
```

4. History of change

Ref.	Date	Title / Version	Responsible
/1/	2004-02-17	Initial Version	Cuijpers
/2/	2004-02-26	Described remaining TODOs (DeleteTables,SetVoicePromptPathSettings,PhoneServices, ProxyServerCommand). Left open: ShowBillingRecords (awaiting Designdecission Gruber G.)	Schweigh
/3/	2004-03-23	Added: Gateway services + modif. in createUser/Domain Added: ProviderLocationAttributes	Schweigh
/4/	2004-05-05	Added: ChangeGatewayLocationForUser, CreateVioceMailDirectoryForUser	Schweigh
/5/	2004-05-14	Added: SshOnAccess	Schweigh
/6/	2004-06-30	Added: A.A. CheckAddressBook, CheckIADForUsers Services: Create/DeleteAddressBook, Create/DeleteIADForUsers	Schweigh
/7/	2004-12-06	Added Services: Add/Delete/ShowAnnouncementsforDomain ModifyUserDetailedData Add/DeleteAttributesForDomain Set/DeletePhoneSettings Changed: Create/ModifyDomain (Licensing) ShowDomainDetailedData ResetPassword (Email+Reset login counter) Login (Login counter) DeleteDomain (delete domain_attr, phones_attr, call ClearDomain)	Schweigh
/8/	2005-02-15	Added Services:	Schweigh

		ShowAnnouncementsUsageForDomain Changed: DeleteAnnouncementsForDomain	
/9/	2005-03-03	Added: Service TriggerSyncServer	Schweigh
/10/	2005-06-07	Added: Service SendEMail Changed: Service ShowGatewayLocationDetails ListDomainUsers	Schweigh
/11/	2005-09-02	Added: Generic DB service Add/Delete/ShowFeatures, Add/Delete/SetProfile, Add/Delete/Set/ShowProfilesForDomain, ChangePhoneSettingsKey	Schweigh

Version: Version 2.1
Date: 2005-09-02
Name:
MISSISSIPPI Management Service descriptions.doc

Author: Cuijpers
Proved:
Released: