Introduce Documentation Version 1.1 Technical Guide

Contents

The latest version of this document can be found at:

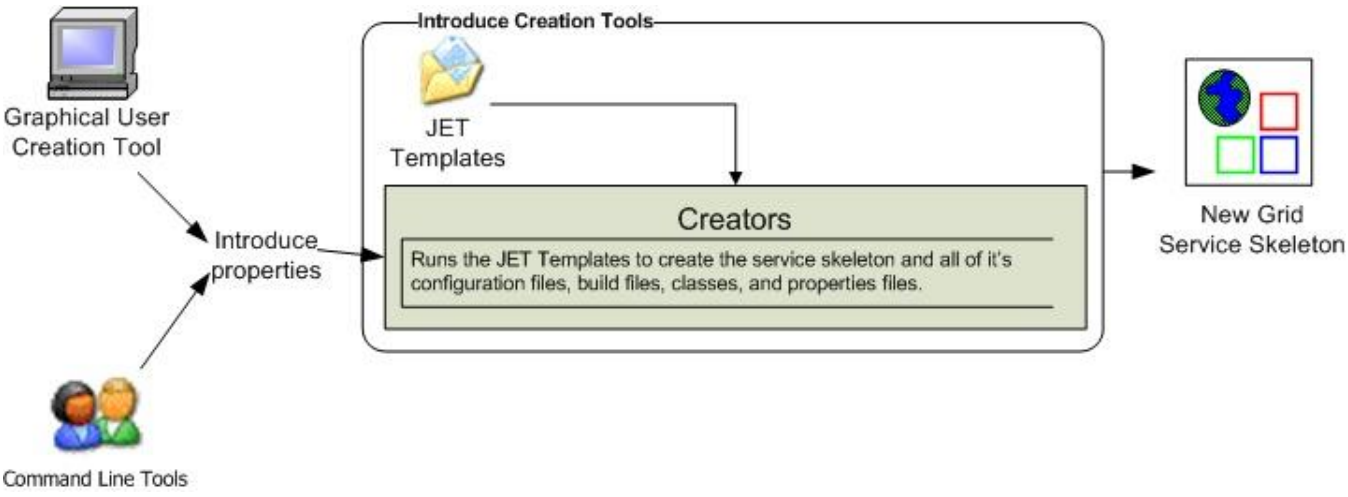
http://www.cagrid.org/mwiki/index.php?title=Introduce:Documentation_Version_1.1_TechnicalGuide

Introduce Design

The runtime support to enable service creation, modification, and deployment is provided by the Introduce engine. In this section, we describe the main components of this engine.

Service Creator

The service creator is composed by a series of templates using the Java Emitter Templates (JET) component, which is part of the Eclipse Modeling Framework (http://www.eclipse.org/emf/), for generating source code and configuration files, and a skeleton set of directories which is used to generate a Grid service that can be built, registered, and deployed in the Grid environment.

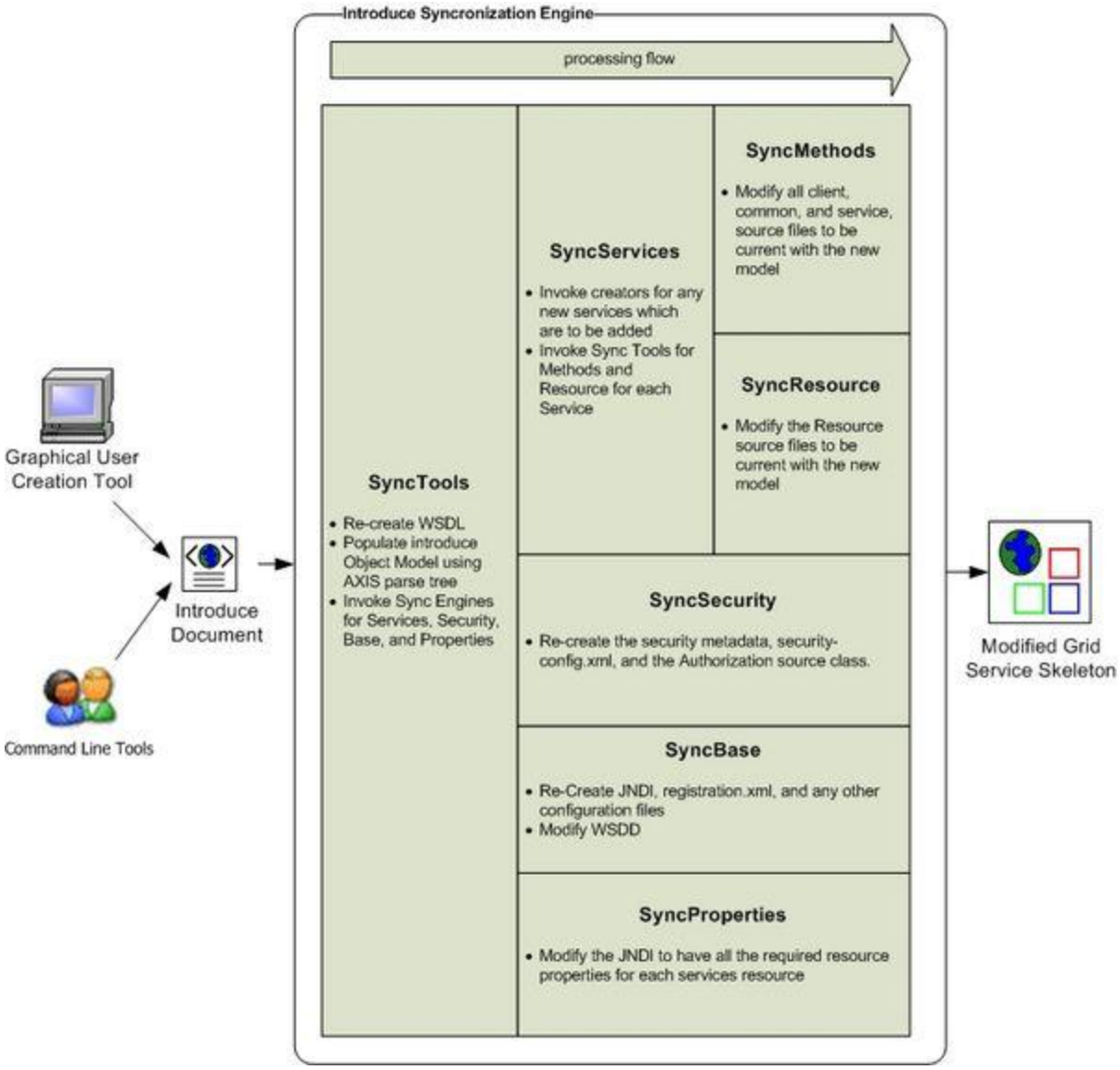


Introduce Creation Framework

Templates for source code and configuration files are used to create all the custom Java source code for the service and the client APIs, and to generate the files required by the GT in order to build and deploy a Grid service. Deployment configuration files are used for resource and resource property configuration in the form of Java Naming and Directory Interface (JNDI), resource property registration configuration, service deployment descriptor in the form of Web Service Deployment Descriptor (WSDD), and security configuration. The basic service created by Introduce has the following components:

- Ant processes for build, deploy, and test operations
- custom configuration files for IDE integration, e.g., Eclipse project files for editing of the service using the Eclipse platform (www.eclipse.org)
- standard interface for both client and service to implement
- fully implemented client APIs
- stub implemented service
- configuration to support service metadata and resource properties and the registration of metadata and properties
- configuration for secure service deployment and authorization

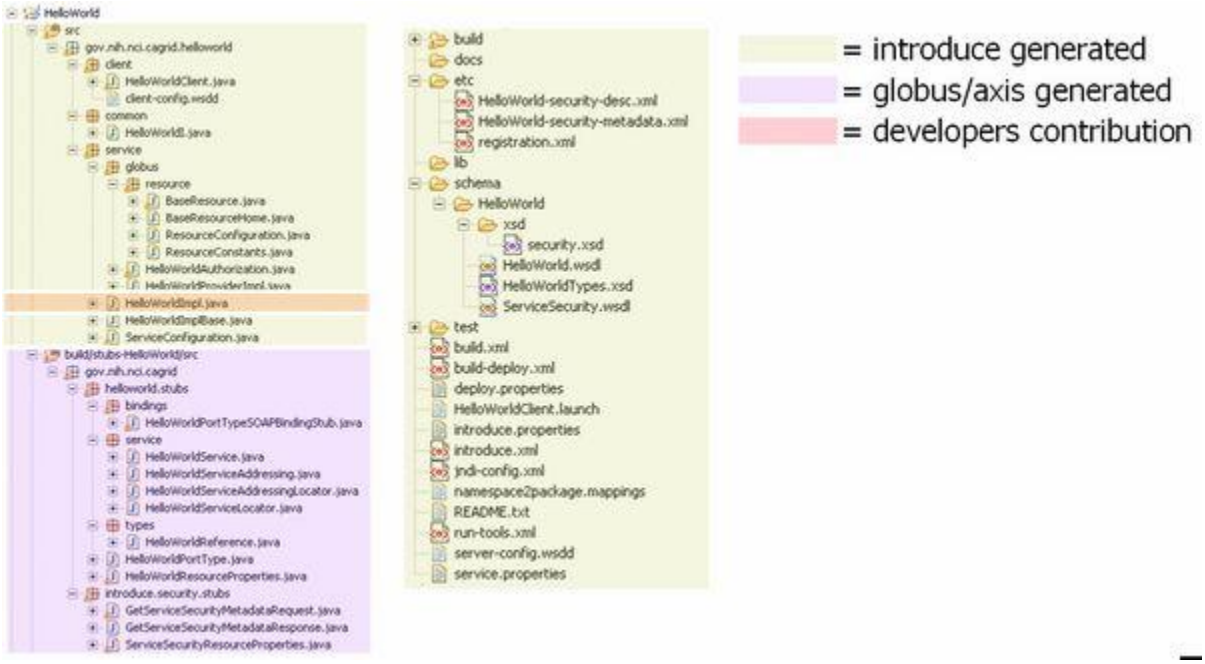
Service Synchronizer



Introduce Synchronization Framework

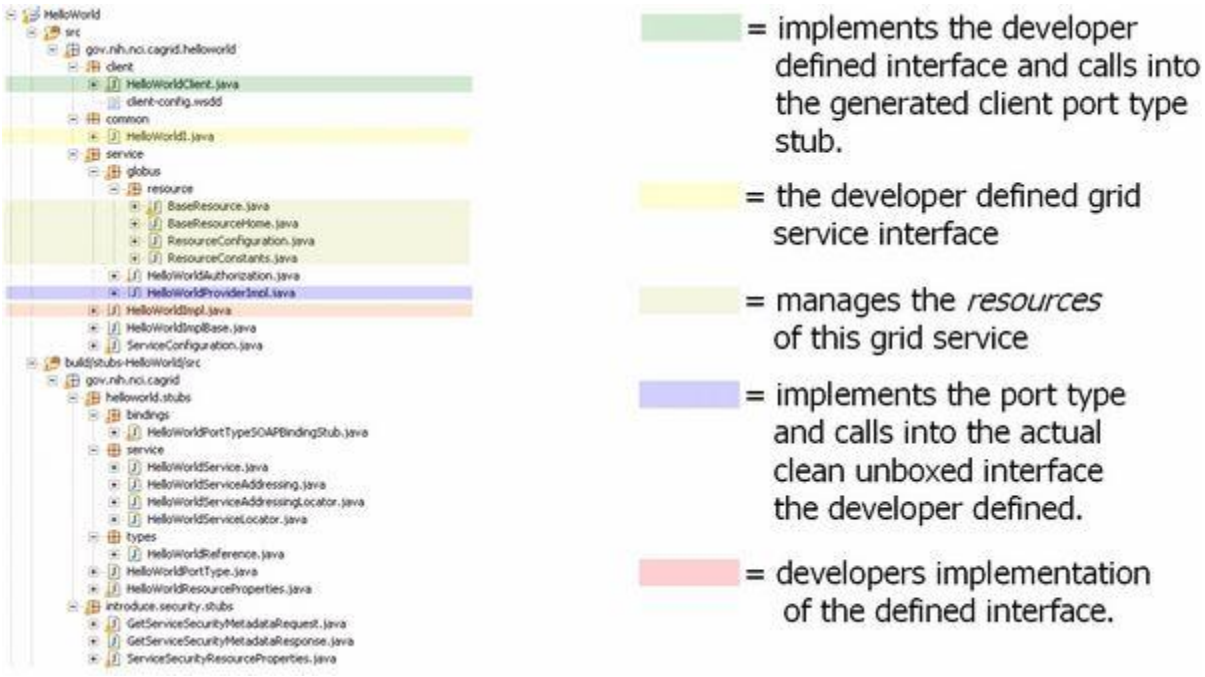
Service resynchronization is the process by which the source code and configuration generation tools of the Introduce toolkit will analyze the service's current implementation with that of the desired service description. This process will add, remove, and modify any service methods, resource properties, and service settings which have been added, removed, or modified from the service description. The descriptions and configurations for methods, metadata, and security are those that are generated from the GDE, programmatically or by hand, and that can be validated by the Introduce service schema[1]. The service description is the basis by which the code generation tools add, remove and modify operations, metadata, and change the security configuration of the service by editing the source code, configuration files, and metadata files. This is similar to the way that Axis will use the WSDL to generate the grid stubs of the service. The overall high level process of service resynchronization is illustrated below.

In this process, modified documents created by the GDE are processed using JET templates to create the modified service skeleton. The service synchronizer component manages the service WSDL description. If a service or method has been added or removed, the respective WSDL files must be updated to reflect the changes. Updating the files requires many auto generated code segments. External XML schemas, which describe published data types, must be imported into the respective WSDL files so that the data types can be located and used to generate the required Java beans and SOAP bindings. The WSDL description is basically the Grid layer representation of the Java based service interface. Both the message types, which represent the data types of the input parameters and the return type of a service method, and the complete operation, which reflects the Java based signature of the method being exposed, have to be described in the WSDL file. The synchronization operation automatically keeps this file in sync during the development of services so that the service implementation and the Grid service description represented by the WSDL match up. This ensures that the methods implemented in the Grid service can actually be invoked.



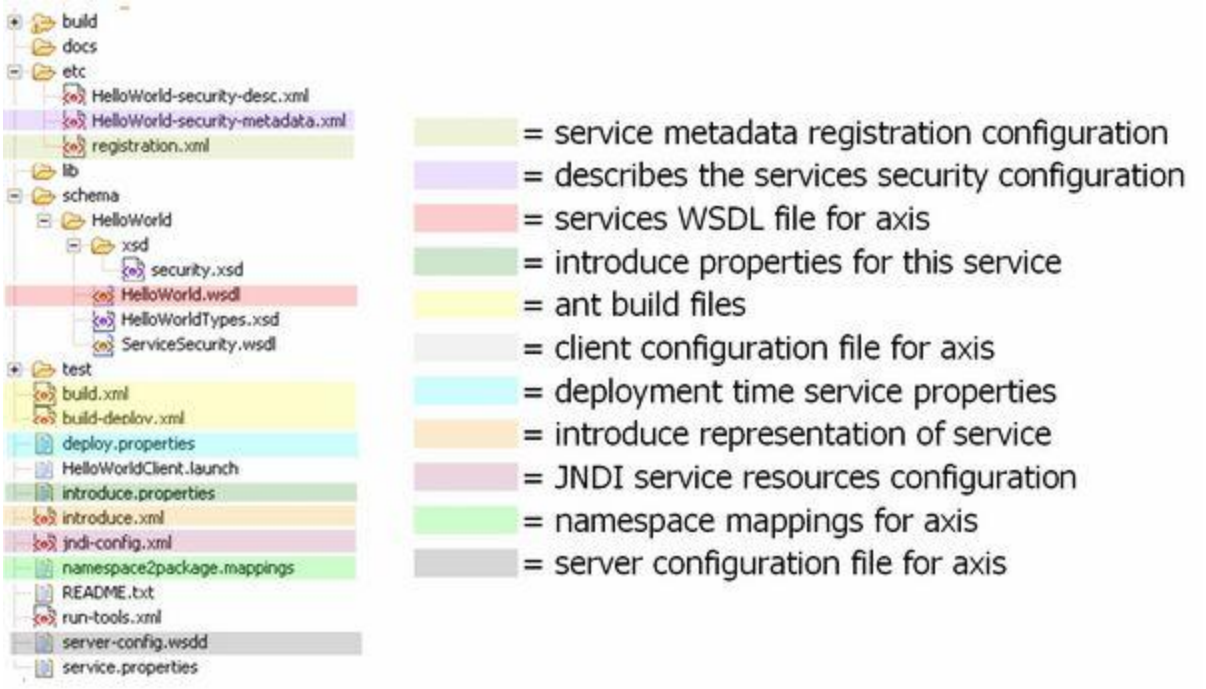
Introduce Generated Skeleton

The figure above shows a basic service layout as created by the Introduce Service Creator component. It shows which pieces of this example service are generated by the code generation tools, which pieces are built by the underlying Globus/Axis tools, and which pieces are to be implemented by the service developer.



Introduce Generated Grid Service Source Code

The figure above describes what some particularly critical source files are generated for in an example service created by the Introduce engine. When a WSDL file is parsed by the Axis engine a PortType interface is created which is the Java representation of the API of the grid service. The Axis generated PortType interface must then be implemented on the service to provide the services implementation. In order to enable the service developer to implement a cleaner, non-documument literal interface, Introduce will automatically create the implementation of this PortType interface (HelloWorldProviderImpl). The Introduced generated implementation of this interface will unbox the document literal calls ot the service and pass them on to the unboxed/clean interface (HelloWorldI) which the user defined. Introduce will generate a stubbed implementation of this interface (HelloWorldImpl) which the user will be responsible for implementing. This class will maintain the services implementation of the methods. This enables the service designer to be shielded from the details of the Axis document literal grid service interface and enable them to implement an interface which is as then originally described. The figure also illustrates the example service's resource and resource home, which are generated to manage the service resource and the service resource properties, respectively.



Introduce Generated Configuration File

The figure above shows the use of the different common files of an example Introduce created service. It shows the files used for configuring registration and security for the grid service, as well as those used by Introduce for synchronization, and those used for build and deployment. This example service created by Introduce also contains Eclipse project files so that the service can easily be edited using the Eclipse platform (www.eclipse.org). The service can have an inheritance model by adding methods from another service; possibly along with the implementations of those methods (see Section 0). If a method were imported from another Grid service, the service synchronizer component would also pull in the WSDL description of the method and copy it into the portType of the new service. This enables the service to have a completely protocol compatible implementation of the method.

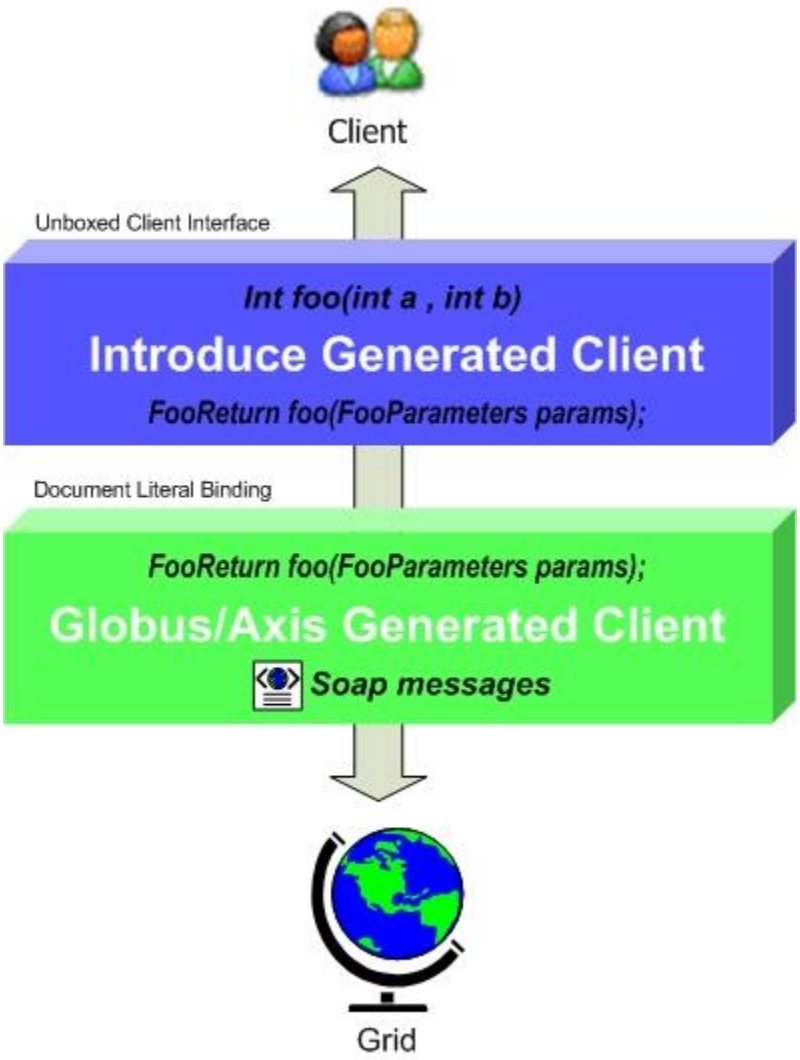
Service Deployer

The service deployment features of Introduce currently support deploying Grid services to a Globus or Tomcat container. The deployment framework can easily be extended to provide deployment capability to other Web Service containers. It utilizes the layout of the Grid Archive (GAR) structure to organize and package a deployable service. Once the GAR for a service is generated by the deployment framework, the GAR can be utilized to deploy the service to the appropriate container. The deployment framework also allows for deployment time service properties to be acquired from the application developer or the user deploying the service. These custom service properties allow a service creator to define configurable options that make sense for a particular deployment of the service to have control over. When the service is deployed, the deployer prompts for the desired values, presenting service developer provided defaults. The supplied values are then made available to the service at runtime. For example, if a service

implementation accesses a local database, the username and password of the database can be specified as service properties. This allows a very convenient way for those deploying the service to configure it appropriately for their environment, without requiring knowledge of how the grid service implements these configuration points. The deployment process first populates files required in deployment (e.g., the WSDD and JNDI of the service) to specify such values as service deployment path or security configuration file locations. It then gathers the library files required for the service as well as those library files which contain the actual runtime code of the service. All configuration files and service resources are also collected. Finally, a GAR file is generated for this service. Once the GAR file has been generated, it can be handed off to the particular deployment handler for the required container to be used for this deployment.

Other Features of the Introduce Engine

Auto-Boxing/Unboxing of Service Operations



Introduce Generated Client Layers

Grid service code generation and synchronization is one of the most complicated code generation operations in the toolkit. This component has to re-write the basic source code of the service so that the client and the server both agree on the newly modified interface as designed by the service creator. It also has to deal with associating the server side implementation of the service designer's interface to the actual port type generated by Globus via Axis. The overall process is complicated due to fact that GT/Axis use document literal bindings to create the services portType and bindings to SOAP. For example, if a user describes a new method as shown below: *int foo(int bar1, int bar2);*

The port type method that will be created for the corresponding Java interface call will look like the one below: *FooReturn foo(FooParameters params);*

This style is known as document literal binding. This boxing or wrapping of the parameters and the return type of the service method can be confusing to the service user and service developer to deal with, especially since this document literal style is exposed directly through the client API or the service implementation API. Every client using the service will have to box up the parameters to call the operations of the service and un-box the results. Not only will this task be cumbersome for service users, but the document literal interface is not the interface that the service designer intended to be provided to its users. The Introduce toolkit will hide the boxing and un-boxing of methods by providing an interface to the service, which looks exactly as described by the service developer and not as interpreted by Globus via Axis.

In order to do this, the toolkit creates a wrapping layer in the client and service which both implement the clean interface (non document literal). These wrapper layers auto-box and un-box and map the calls from the clean client to the document literal port type client generated by Globus/Axis and visa versa for the service. It is worth noting that the move to document literal binding was made in the GT for interoperability reasons, as the more developer-friendly APIs provided by Introduce, and previously provided by GT, are not standardized and that Introduce created services can still be accessed via the standard document literal interfaces.

Resource Framework

Introduce provides support for exposing service state and metadata by abstracting away the details of common patterns of use of the WSRF. The WSRF specifications and the GT implementation of these specifications allow the creation of stateful Grid services, whilst remaining compatible with existing Web Service standards. A Grid service's state (and metadata) is maintained in the Resource represented by the service, as described in the WSRF specification, and is exposed to clients by way of Resource Properties. Introduce allows its users to expose state or metadata of their services by managing the definition, creation, and population of the Resource Properties of a service instance. Upon creating a service, a service developer is able to simply select from a list of common resource usage patterns, and Introduce will manage the complete generation of the necessary backend code and configuration to implement that pattern. Once the resource pattern is defined, a service developer can then select data types they wish to use in order to expose state or metadata of its resources by way of Resource Properties.

Developers need only select a data type they wish to expose, and Introduce makes it available as a WSRF standard Resource Property. Developers are able to either programmatically control the values of the property at runtime (as is common for exposed resource state), or supply the initial value from a file at service startup (as is common for exposed metadata). Another powerful feature of Resource Properties is that they can be advertised to, and aggregated at, a remote indexing service – an indexing service can be used by clients and other services to discover services in the Grid environment. GT provides such a service in the form of the Index Service. It acts as the white and yellow pages of the grid, providing resource and service discovery. Introduce allows a service developer to maintain soft-state registration of its resource properties by simply selecting a check box on each property they wish to register. For each such property, it creates all of the necessary source code and configuration changes necessary to periodically register the property with an Index Service, and ensure the most recent value of the property is made available in the Index Service and associated with the service. The combination of these two simple-to-use, yet powerful features, allows a service provider to, for instance, automatically provide and register a description of its service to a central repository when the service is running.

Compositional Inheritance

Introduce enables the service developer to import methods from other services (i.e., from other port types). This notion is called compositional inheritance. Services under the new WSDL 2.0 specification cannot extend *portType* definitions. In order to simulate *portType* extension Introduce implements the ability to copy operation descriptions from other port types and put them in the service's own description. When using this feature, there are various configuration option the engine must know such as the namespaces of the operations, and whether or not an implementation of the operations are already provided or should be generated and stubbed just as any other Introduce defined operation. In order to be sure that the operation can be correctly imported and copied into the new port type, the WSDL of this operation and any referenced schemas must be brought into this new service and imported. If the operation implementation is not being provided, the synchronization engine must add this operation to the services base interface, and the un-boxing Globus wrapper for this operation must be generated. If the implementation of this operation is being provided, the extra code in the interface, its implementation, and the wrapper do not need to be added. However, the implementation code, in the form of a JAR file, must be brought into the new service, and the operation class must be added to the WSDD of this service.

Service Contexts

In grid services architecture it is sometimes required that a service not only maintain some extra information about the service but also maintain information (state) which is of particular interest to one particular user. These styles of grid service use cases have driven the requirements for specifying a mechanism for stateful Grid services. Each WSRF service manages its state by creating and manipulating Resources. A Resource can essentially be thought of as an arbitrary state representation, as defined by the service developer. The main restriction is that a given service can only manage a single given resource type. As previously stated in section 0, Introduce supports this mechanism through the Service Context concept. For example, a Data

Service may want to define two contexts: the main "Query" context which provides the ability to query into a backend database, and the secondary "Results Delivery" context which provides the ability to iteratively access query results (similar to a remote cursor). Introduce implements these Service Contexts by creating a WSRF service for each context, and a corresponding Resource type. In this example, the Query Service will have a Resource type that represents the backend database(s), and the Results Delivery Service would have a Resource type that represents query results. In this design, when a client invokes the query operation on the Query Service, the service can query the database and then create an instance of the Results Delivery Service's Resource. The query operation then returns a pointer, or EndPointReference (EPR), to this Resource. The client is then able to interact with the query results by using the Results Delivery Service's client, passing in the returned EPR to its constructor. Through its support for multiple Service Contexts, Introduce enables this and other such Resource patterns for stateful grid services. This support is managed by the synchronization component of the Introduce engine. The service description model, created via the GDE or programmatically/hand generated, enables the description of multiple Service Contexts in an Introduce service. Each Introduce service has at least one Service Context (the main service), and can create an arbitrary number of additional Service Contexts to support more complex resource usage patterns, as described above. Each created context has a corresponding source directory containing its own server, client, common, resource, etc. This enables resource properties, operations, and security configurations to be added, removed, and modified for each additional context. Each additional context's corresponding service and resource are modified, compiled, and deployed with the main service.

Security

Introduce facilitates the creation and configuration of secure Grid services using the Grid Security Infrastructure (GSI) and allows security to be configured at both the service level and the service method level. Moreover, Security can be enforced on both the client and service sides. Introduce allows service developers to specify the secure communication mechanism(s), in which clients are allowed to communicate with the service. An Introduce generated client can automatically be configured to communicate over the secure communication mechanism specified by the service. In the case where multiple secure communication mechanisms are supported by the service, Introduce will allow the service developer to choose which mechanism the client will use.

Grid security can be complex, and a detailed discussion on it is out of the scope of this document. The Globus documentation (http://www.globus.org/toolkit/docs/4.0/security/) and tutorials (http://gdp.globus.org/gt4-tutorial) provide a good deal of overview and details on the topic. For the most part, Introduce users need not concern themselves with all the details, but should have a basic understanding of what is happening under the hood, and should understand how provide credentials for secure communication with services.

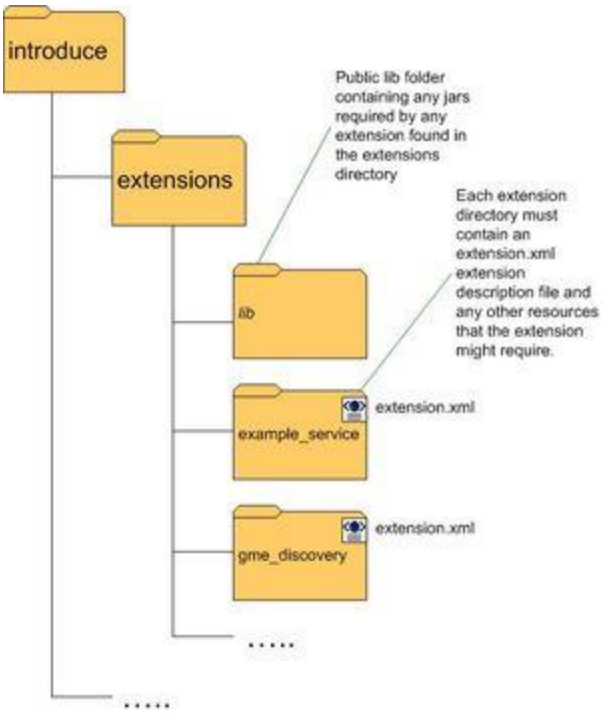
Introduce builds on GSI (Globus Security Infrastructure), and hence uses Public Key Cryptography (PKI). Both services and clients may optionally have credentials (certificates), and authenticate and authorize each other. Services and corresponding clients generated from the Introduce toolkit attempt to automatically configure security appropriately, and this behavior is

sufficient for most users, however it can be overridden (either by manually configuring this client "stub" or by overriding the provided configureStubSecurity method). Introduce clients will attempt to communicate anonymously with services, as long as the service allows it (as advertised via its ServiceSecurityMetadata). If a service does not allow anonymous communication, client credentials must be used to authenticate with the service. Introduce created clients will attempt to use the default Globus credentials if present (via a grid-proxy-init, or logging in with Dorian and specifying to set the credentials as the defaults). Alternatively, Introduce created clients have constructors which take credentials (GlobusCredential), and also have an appropriate setter method (setProxy), which can be used after construction. Some services have different security requirements for different operations, so it may not be immediately obvious whether or not credentials are required. As such, when communicating with secure services it is good practice to have a valid grid proxy set as default, or specified on the client; the client APIs will only use it if necessary.

Authorization

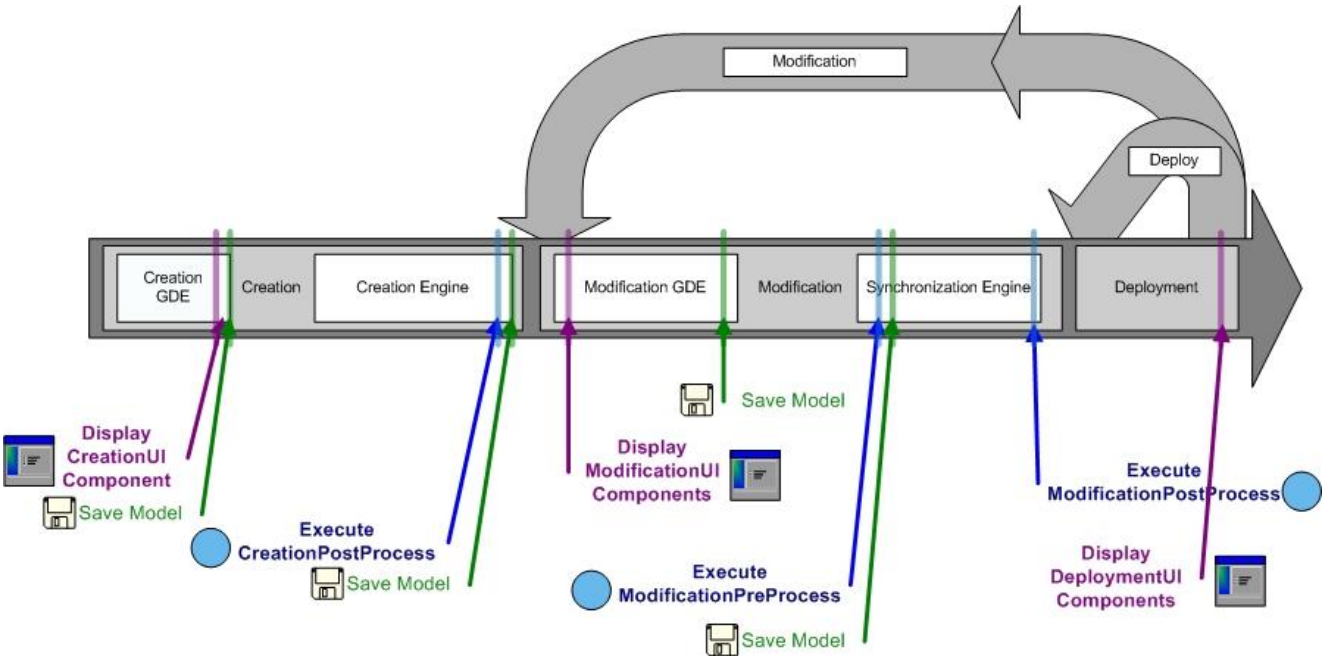
Introduce uses the Globus PDP infrastructure for authorization. The PDP framework enables a class or series of classes which implement the PDP or PIP interfaces to be chained together. This chain of authorization classes can be added at the service, method, or resource level. These classes will loaded and called serialy each time the service is invoked and the service descriptor indecates the chain needs to be used. The classes will each be able to accept or reject the call to the method based on the credientials of the caller or any other information they wish to recieve from any place else. Introduce will automaticaly generate the PDP class to support authorization based on GridGrouper and/or CSM. This class will automatically be generated by the Synchronization Engine and written to <service package name>.service.globus.Authorization.java. The user also has the ability to put in a custom PDP chain. The Introduce enables this and when this is the case Introduce will write there specified PDP chain string into the security-descriptor.xml for the servce instead of the one that Introduce generates. For more detail on the service side security configuration of a globus service please see the Globus documentation.

Introduce Extension Framework



The Introduce Extension framework currently consists of two styles of extensions; service and data type discovery. These extensions are implemented by service or data type discovery components which add custom functionality to the Introduce framework. Extensions are added to the toolkit in the form of extension plug-ins, which the toolkit will then be able to expose to the user. To provide an extension to Introduce, the extension provider must implement or extend the appropriate classes for the style of extension they wish to provide, and must fill out the extension XML configuration document. Once this extension is implemented and configured it can be placed in the extensions directory of Introduce. This directory has a common library (lib) area which enables it to avoid using custom class loaders for each extension.

Service Extensions



Introduce Service Extension Execution Timing

A service extension is one which enables customization of the service creation and modification processes. These extensions can add required operations, service resources or resource properties, or security settings, for example. The service extension allows the user to provide custom code that will be executed at different times throughout the creation and modification processes of service development. A service extension consists of 6 main extension components that can be implemented and provided by the developer: CreationPostProcess, CreationUIDialog, CodegenPreProcess, CodegenPostProcess, ServiceModificationUIPanel, and ServiceDeploymentUIPanel. Each of these extension components have a predefined class and interface that must be extended or implemented. Three of the service extension components, CreationUIDialog, ServiceModificationUIPanel, and ServiceDeploymentUIPanel are graphical components provided to the Introduce GDE, and the other three are Introduce engine plug-ins.

Each service extension component is invoked at a specific point in the creation or modification steps. These different time points for each component execution are critical for making certain changes. For example, when the CreationUIDialog component for a particular extension is executed, the service has been created as a blank service and no modification or synchronization has been done on the service. At this point the CreationUIDialog might prompt the user for particular information about the creation processes. The CreationUI component would only be executed/displayed once for any given service and its non graphical component, the CreationPostProcess, will also only be ran one time after the service has been created and before it will ever be modified. The modification components are ran every time a service is saved and synchronized. The graphical modification component is always available in the Introduce GDE during modification time. The two service modification engine components, ModificationPreProcess and ModificationPostProcess are executed respectively, before and after the synchronization process is executed. This enables ModificationPreProcess to do such things as modify the ServiceDesription, represented by the Introduce service description file, or the

WSDL files of the services. The ModificationPostProcess, on the other hand, might move in required files, or populated stubbed methods, etc.

Below is an example *extension.xml* for a service extension. You can see that this extension is a SERVICE type extension and that it is version 1.1 of the extension. This version is the version of the extension and not of the Introduce that it works with. You can also see that this extension is called *data* and is displayed as *Data Service*. Three is also a list of the extension components and the classes that will be used to invoke them. Each component style must extend or implement from a specific base class or interface in introduce. Each of these is optional, however, this extension is using all components of the service extension framework. This extension also has some configuration propreties; one for the address of the Global Model Exchange and one for the address of the caDSR. This extension also provides an upgrader. It is showing that this upgrader can be ran to move any service with a data extension version of *null* to be able to be managed by this version of the extension which is 1.1. This will be used by the service migration framework when upgrading service from older versions of introduce or extension to newer versions available.

```
<ns1:ExtensionDescription
  extensionType="SERVICE"
  version="1.1"
  xmlns:ns1="gme://gov.nih.nci.cagrid.introduce/1/Extension">
  <ns1:ServiceExtensionDescription
    displayName="Data Service"
    name="data">
    <ns1:CreationPostProcessor>

gov.nih.nci.cagrid.data.creation.DataServiceQueryOperationProviderCreator
    </ns1:CreationPostProcessor>
    <ns1:CreationUIDialog>
      gov.nih.nci.cagrid.data.ui.creation.DataServiceCreationDialog
    </ns1:CreationUIDialog>
    <ns1:CodegenPreProcessor>
      gov.nih.nci.cagrid.data.codegen.DataServiceCodegenPreProcessor
    </ns1:CodegenPreProcessor>
    <ns1:CodegenPostProcessor>

gov.nih.nci.cagrid.data.codegen.DataServiceOperationProviderCodegenPostProces
sor
    </ns1:CodegenPostProcessor>
    <ns1:ServiceModificationUIPanel>
      gov.nih.nci.cagrid.data.ui.DataServiceModificationPanel
    </ns1:ServiceModificationUIPanel>
    <ns1:ServiceDeploymentUIPanel>
      gov.nih.nci.cagrid.data.ui.auditors.AuditorDeploymentConfigPanel
    </ns1:ServiceDeploymentUIPanel>
    <ns1:Properties>
      <ns1:Property
        makeGlobal="true"
        type="SERVICE_URL"
        key="Global Model Exchange URL"
        value="http://cagrid01.bmi.ohio-
state.edu:8080/wsrf/services/cagrid/GlobalModelExchange" />
      <ns1:Property
```

```
        makeGlobal="true"
        type="SERVICE_URL"
        key="Cancer Data Standards Repository (caDSR) URL"
        value="http://cagrid05.bmi.ohio-
state.edu:8080/wsrf/services/cagrid/CaDSRService" />
    </ns1:Properties>
  </ns1:ServiceExtensionDescription>
  <ns1:UpgradesDescription>
    <ns1:UpgradeDescription

upgradeClass="gov.nih.nci.cagrid.data.upgrades.DataServiceUpgrade1pt0to1pt1"
      toVersion="1.1" />
  </ns1:UpgradesDescription>
</ns1:ExtensionDescription>
```

Data Type Discovery Extensions

These extensions are Introduce GDE components that will be available at the service modification step. These components are intended to be able to provide custom data type discovery for the service developer. The custom data type discovery component must allow the developer to browse types and chose to use those types in the developed service. This means that the data type discovery extension will have to be able to copy the schemas which represent the data types down to the service's schema directory and produce a NamespaceType object for the namespace of each separate data type. This enables the grid service to utilize the schemas for describing the data types which are used in the WSDL messages traveling in and out of the created service. In addition to default plug-ins, domain specific plug-ins can be installed in Introduce. For example, in the caBIG environment, a caDSR discovery plug-in is provided with Introduce. This plug-in allows service developers to locate and use data types registered in the cancer Data Standards Repository (caDSR), which is a curated repository of common data elements used in caBIG.

Introduce Model

The Introduce Model is used as the input to the Introduce Synchronization Engine. The model describes all aspects of the service from methods to security constraints. Below is the detailed schema for the model.

gme://gov.nih.nci.cagrid/1/Introduce

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid/1/Introduce"
  xmlns:extension="gme://gov.nih.nci.cagrid.introduce/1/Extension"
  xmlns:service="gme://gov.nih.nci.cagrid.introduce/1/Services"
  xmlns:namespace="gme://gov.nih.nci.cagrid.introduce/1/Namespace"
  xmlns:property="gme://gov.nih.nci.cagrid.introduce/1/Property"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="gme://gov.nih.nci.cagrid/1/Introduce"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xs:import
```

```
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Namespace"
    schemaLocation="./Namespace.xsd" />
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Services"
    schemaLocation="./Services.xsd" />
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Extension"
    schemaLocation="./Extension.xsd" />
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Property"
    schemaLocation="./Property.xsd" />
  <xs:element name="ServiceDescription">
    <xs:complexType>
      <xs:sequence>
        <xs:annotation>
          <xs:documentation>
            Namespaces represent any available type which can be used
            for any purpose in this service. Services represent any
            service present in this grid service. The first service in
            the list is the primary service. other services are used to
            prepresent other resource types this service may be able to
            create. Extensions describe the extensions that this
            particular service needs to be executed while creating
            and/or building this service. oder matters on the
            extensions. ServiceProperties are properties which will be
            configured at deploy time and available to the service
            implementation to be used for configuration.
          </xs:documentation>
        </xs:annotation>
        <xs:element
          ref="namespace:Namespaces"
          minOccurs="1"
          maxOccurs="1" />
        <xs:element
          ref="service:Services"
          minOccurs="1"
          maxOccurs="1" />
        <xs:element
          ref="extension:Extensions"
          minOccurs="0"
          maxOccurs="1" />
        <xs:element
          ref="property:ServiceProperties"
          minOccurs="0"
          maxOccurs="1" />
      </xs:sequence>
      <xs:attribute
        name="introduceVersion"
        type="xs:string"
        use="optional" />
      <xs:attribute
        name="description"
        type="xs:string"
        use="optional" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Namespace"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Namespace"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:attributeGroup name="BasicSchemaDervivedType">
    <xs:attribute
      name="type"
      type="xs:string"
      use="required">
      <xs:annotation>
        <xs:documentation>
          Indicates the QName of the metadata.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute
      name="packageName"
      type="xs:string"
      use="optional">
      <xs:annotation>
        <xs:documentation>
          Indicates a posible packageName change from parent
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:attributeGroup>
  <xs:attributeGroup name="CustomSchemaDervivedType">
    <xs:attributeGroup ref="BasicSchemaDervivedType" />
    <xs:attribute
      name="className"
      type="xs:string"
      use="optional">
      <xs:annotation>
        <xs:documentation>
          Used to specify the class name of an existing object that can
          serialize to, and deserialize from, the document defined by
          the namespace:type attributes.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute
      name="serializer"
      type="xs:string"
      use="optional">
      <xs:annotation>
        <xs:documentation>
          The Axis serializer factory class to use
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute
```

```
    name="deserializer"
    type="xs:string"
    use="optional">
    <xs:annotation>
      <xs:documentation>
        The Axis deserializer factory class to use
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:attributeGroup>
<xs:element
  name="Namespaces"
  type="NamespacesType">
  <xs:annotation>
    <xs:documentation>
      List of methods for this interface.
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="NamespacesType">
  <xs:sequence>
    <xs:element
      name="Namespace"
      type="NamespaceType"
      maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="NamespaceType">
  <xs:sequence>
    <xs:element
      name="SchemaElement"
      type="SchemaElementType"
      minOccurs="0"
      maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute
    name="namespace"
    type="xs:string"
    use="required">
    <xs:annotation>
      <xs:documentation>
        Used to specify the package to be use when the object is
        created of by use of an already existing object.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute
    name="packageName"
    type="xs:string"
    use="optional">
    <xs:annotation>
      <xs:documentation>
        Used to specify the package to be use when the object is
        created of by use of an already existing object.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
```

```
    <xs:attribute
      name="location"
      type="xs:string"
      use="optional">
      <xs:annotation>
        <xs:documentation>
          Used to specify the location schema or wsdl on the local
          filesystem. If it is omitted, it is assumed it can be
          retrieved from a GME.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute
      name="generateStubs"
      type="xs:boolean"
      use="optional">
      <xs:annotation>
        <xs:documentation>
          Used to specify whether or not stubs/beans should be generated
          for the given namespace. If omiited, "true" is assumed. If set
          to "false" no beans will be created for any of the types
          defined in the corresponding schema, nor any of the types in
          any schemas imported by this schema.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="SchemaElementType">
    <!-- Stupid xsd won't let you have a choice of attribute groups,
      so need to be of the cutstom type and make the added attributes all
optional
      (when they should all be required).
      What I'd really like here is a choice between Basic and Custom -->
    <xs:attributeGroup ref="CustomSchemaDervivedType" />
  </xs:complexType>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Services

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Services"
  xmlns:tns="gme://gov.nih.nci.cagrid.introduce/1/Services"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:methods="gme://gov.nih.nci.cagrid.introduce/1/Methods"
  xmlns:security="gme://gov.nih.nci.cagrid.introduce/1/Security"
  xmlns:resources="gme://gov.nih.nci.cagrid.introduce/1/Resources"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Services"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Methods"
    schemaLocation="./Methods.xsd" />
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Resources"
    schemaLocation="./Resources.xsd" />
```

```
<xs:import
  namespace="gme://gov.nih.nci.cagrid.introduce/1/Security"
  schemaLocation="./Security.xsd" />
<xs:element
  name="Services"
  type="tns:ServicesType">
  <xs:annotation>
    <xs:documentation>
      List of Services for this Grid Service. The first service in the
      list is the primary service for this grid service. Others
      represent resource types which can be created by this service.
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="ServicesType">
  <xs:sequence>
    <xs:element
      name="Service"
      maxOccurs="unbounded"
      type="tns:ServiceType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceType">
  <xs:sequence>
    <xs:element
      ref="methods:Methods"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element
      ref="resources:ResourcePropertiesList"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element
      ref="security:ServiceSecurity"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element
      name="description"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1" />
  </xs:sequence>
  <xs:attribute
    name="name"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="namespace"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="packageName"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="resourceFrameworkType"
    type="xs:string"
```

```
      use="required" />
  </xs:complexType>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Methods

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Methods"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:security="gme://gov.nih.nci.cagrid.introduce/1/Security"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Methods"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import
    namespace="gme://gov.nih.nci.cagrid.introduce/1/Security"
    schemaLocation="./Security.xsd" />
  <xs:element
    name="Methods"
    type="MethodsType">
    <xs:annotation>
      <xs:documentation>
        List of methods for this services interface.
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="MethodsType">
    <xs:sequence>
      <xs:element
        name="Method"
        minOccurs="0"
        maxOccurs="unbounded"
        type="MethodType" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MethodType">
    <xs:sequence>
      <xs:element
        name="Inputs"
        minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element
              name="Input"
              minOccurs="0"
              maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute
                  name="qName"
                  type="xs:QName"
                  use="required" />
                <xs:attribute
                  name="isArray"
                  type="xs:boolean"
                  use="required" />
```

```
            <xs:attribute
              name="name"
              type="xs:string"
              use="required" />
            <xs:attribute
              name="containerClass"
              type="xs:string"
              use="optional" />
            <xs:attribute
              name="description"
              type="xs:string"
              use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Output">
    <xs:complexType>
      <xs:attribute
        name="qName"
        type="xs:QName"
        use="required" />
      <xs:attribute
        name="isArray"
        type="xs:boolean"
        use="required" />
      <xs:attribute
        name="isClientHandle"
        type="xs:boolean"
        use="optional" />
      <xs:attribute
        name="clientHandleClass"
        type="xs:string"
        use="optional" />
      <xs:attribute
        name="isCreatingResourceForClientHandle"
        type="xs:boolean"
        use="optional" />
      <xs:attribute
        name="resourceClientIntroduceServiceName"
        type="xs:string"
        use="optional" />
      <xs:attribute
        name="description"
        type="xs:string"
        use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:element
    name="Exceptions"
    minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Exception"
          minOccurs="0"
```

```
        maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute
            name="name"
            type="xs:string"
            use="required" />
          <xs:attribute
            name="qname"
            type="xs:QName"
            use="optional" />
          <xs:attribute
            name="description"
            type="xs:string"
            use="optional" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element
  ref="security:MethodSecurity"
  minOccurs="0"
  maxOccurs="1" />
<xs:element
  name="ImportInformation"
  minOccurs="0"
  maxOccurs="1">
  <xs:complexType>
    <xs:attribute
      name="portTypeName"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="namespace"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="packageName"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="wsdlFile"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="fromIntroduce"
      type="xs:boolean"
      use="optional" />
    <xs:attribute
      name="inputMessage"
      type="xs:QName"
      use="optional" />
    <xs:attribute
      name="outputMessage"
      type="xs:QName"
      use="optional" />
  </xs:complexType>
```

```
      </xs:element>
      <xs:element
        name="ProviderInformation"
        minOccurs="0"
        maxOccurs="1">
        <xs:complexType>
          <xs:attribute
            name="providerClass"
            type="xs:string"
            use="optional" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute
      name="name"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="isImported"
      type="xs:boolean"
      use="required" />
    <xs:attribute
      name="isProvided"
      type="xs:boolean"
      use="required" />
    <xs:attribute
      name="inputMessageClass"
      type="xs:string"
      use="optional" />
    <xs:attribute
      name="outputMessageClass"
      type="xs:string"
      use="optional" />
    <xs:attribute
      name="boxedInputParameter"
      type="xs:string"
      use="optional" />
    <xs:attribute
      name="boxedOutputParameter"
      type="xs:string"
      use="optional" />
    <xs:attribute
      name="description"
      type="xs:string"
      use="optional" />
  </xs:complexType>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Property

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Property"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Property"
  elementFormDefault="qualified"
```

```
  attributeFormDefault="unqualified">
  <xs:element name="ServiceProperties">
    <xs:annotation>
      <xs:documentation>
        Properties which will be configurable during deploy time and
        available to the service
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Property"
          minOccurs="0"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute
              name="key"
              type="xs:string"
              use="required" />
            <xs:attribute
              name="value"
              type="xs:string"
              use="required" />
            <xs:attribute
              name="isFromETC"
              type="xs:boolean"
              use="optional" />
            <xs:attribute
              name="description"
              type="xs:string"
              use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Security

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Security"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sec="gme://gov.nih.nci.cagrid.introduce/1/Security"
  xmlns:grouper="http://cagrid.nci.nih.gov/1/GridGrouper"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Security"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import
    namespace="http://cagrid.nci.nih.gov/1/GridGrouper"
    schemaLocation="./gridgrouper/xsd/gridgrouper.xsd" />
  <xs:element
    name="ServiceSecurity"
    type="sec:ServiceSecurity" />
  <xs:complexType name="ServiceSecurity">
```

```
  <xs:sequence>
    <xs:element
      name="SecuritySetting"
      minOccurs="1"
      maxOccurs="1"
      type="SecuritySetting" />
    <xs:element
      name="TransportLevelSecurity"
      minOccurs="0"
      maxOccurs="1"
      type="TransportLevelSecurity" />
    <xs:element
      name="SecureConversation"
      minOccurs="0"
      maxOccurs="1"
      type="SecureConversation" />
    <xs:element
      name="SecureMessage"
      minOccurs="0"
      maxOccurs="1"
      type="SecureMessage" />
    <xs:element
      name="RunAsMode"
      minOccurs="0"
      maxOccurs="1"
      type="RunAsMode" />
    <xs:element
      name="AnonymousClients"
      minOccurs="0"
      maxOccurs="1"
      type="AnonymousCommunication" />
    <xs:element
      name="ServiceCredentials"
      minOccurs="0"
      maxOccurs="1"
      type="ServiceCredential" />
    <xs:element
      name="ServiceAuthorization"
      minOccurs="0"
      maxOccurs="1"
      type="ServiceAuthorization" />
  </xs:sequence>
</xs:complexType>
<xs:element
  name="MethodSecurity"
  type="sec:MethodSecurity" />
<xs:complexType name="MethodSecurity">
  <xs:sequence>
    <xs:element
      name="SecuritySetting"
      minOccurs="1"
      maxOccurs="1"
      type="SecuritySetting" />
    <xs:element
      name="TransportLevelSecurity"
      minOccurs="0"
      maxOccurs="1"
```

```
      type="TransportLevelSecurity" />
    <xs:element
      name="SecureConversation"
      minOccurs="0"
      maxOccurs="1"
      type="SecureConversation" />
    <xs:element
      name="SecureMessage"
      minOccurs="0"
      maxOccurs="1"
      type="SecureMessage" />
    <xs:element
      name="RunAsMode"
      minOccurs="0"
      maxOccurs="1"
      type="RunAsMode" />
    <xs:element
      name="AnonymousClients"
      minOccurs="0"
      maxOccurs="1"
      type="AnonymousCommunication" />
    <xs:element
      name="MethodAuthorization"
      minOccurs="0"
      maxOccurs="1"
      type="MethodAuthorization" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceAuthorization">
  <xs:choice>
    <xs:element
      name="NoAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="NoAuthorization" />
    <xs:element
      name="GridMapAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="GridMapAuthorization" />
    <xs:element
      name="GridGrouperAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="grouper:MembershipExpression" />
    <xs:element
      name="CSMAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="CSMAuthorization" />
    <xs:element
      name="CustomPDPChainAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="CustomPDPChainAuthorization" />
  </xs:choice>
</xs:complexType>
```

```
<xs:complexType name="MethodAuthorization">
  <xs:choice>
    <xs:element
      name="NoAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="NoAuthorization" />
    <xs:element
      name="GridGrouperAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="grouper:MembershipExpression" />
    <xs:element
      name="CSMAuthorization"
      minOccurs="1"
      maxOccurs="1"
      type="CSMAuthorization" />
  </xs:choice>
</xs:complexType>
<xs:complexType name="NoAuthorization" />
<xs:complexType name="GridMapAuthorization">
  <xs:sequence>
    <xs:element
      name="GridMapFileLocation"
      minOccurs="1"
      maxOccurs="1"
      type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="CSMAuthorization">
  <xs:sequence>
    <xs:element
      name="ApplicationContext"
      minOccurs="1"
      maxOccurs="1"
      type="xs:string" />
    <xs:element
      name="ProtectionMethod"
      minOccurs="1"
      maxOccurs="1"
      type="ProtectionMethod" />
    <xs:element
      name="CustomProtectionMethod"
      minOccurs="0"
      maxOccurs="1"
      type="xs:string" />
    <xs:element
      name="Privilege"
      minOccurs="1"
      maxOccurs="1"
      type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="CustomPDPChainAuthorization">
  <xs:sequence>
    <xs:element
      name="PDPChain"
```

```
      minOccurs="1"
      maxOccurs="1"
      type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="ProtectionMethod">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ServiceURI" />
    <xs:enumeration value="ServiceType" />
    <xs:enumeration value="Custom" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="TransportLevelSecurity">
  <xs:sequence>
    <xs:element
      name="CommunicationMethod"
      minOccurs="1"
      maxOccurs="1"
      type="CommunicationMethod" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="SecureConversation">
  <xs:sequence>
    <xs:element
      name="CommunicationMethod"
      minOccurs="1"
      maxOccurs="1"
      type="CommunicationMethod" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="SecureMessage">
  <xs:sequence>
    <xs:element
      name="CommunicationMethod"
      minOccurs="1"
      maxOccurs="1"
      type="CommunicationMethod" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceCredential">
  <xs:choice>
    <xs:element
      name="X509Credential"
      minOccurs="1"
      maxOccurs="1"
      type="X509Credential" />
    <xs:element
      name="ProxyCredential"
      minOccurs="1"
      maxOccurs="1"
      type="ProxyCredential" />
  </xs:choice>
</xs:complexType>
<xs:complexType name="X509Credential">
  <xs:sequence>
    <xs:element
      name="certificateLocation"
```

```
        minOccurs="1"
        maxOccurs="1"
        type="xs:string" />
      <xs:element
        name="privateKeyLocation"
        minOccurs="1"
        maxOccurs="1"
        type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ProxyCredential">
    <xs:sequence>
      <xs:element
        name="proxyLocation"
        minOccurs="1"
        maxOccurs="1"
        type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="SecuritySetting">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None" />
      <xs:enumeration value="Custom" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AnonymousCommunication">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Yes" />
      <xs:enumeration value="No" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="RunAsMode">
    <xs:restriction base="xs:string">
      <xs:enumeration value="System" />
      <xs:enumeration value="Service" />
      <xs:enumeration value="Resource" />
      <xs:enumeration value="Caller" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CommunicationMethod">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Integrity" />
      <xs:enumeration value="Privacy" />
      <xs:enumeration value="Integrity_Or_Privacy" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Resources

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Resources"
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Resources"
  xmlns:tns="gme://gov.nih.nci.cagrid.introduce/1/Resources"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element
  name="ResourcePropertiesList"
  type="tns:ResourcePropertiesListType" />
<xs:complexType name="ResourcePropertiesListType">
  <xs:sequence>
    <xs:element
      name="ResourceProperty"
      type="tns:ResourcePropertyType"
      minOccurs="0"
      maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ResourcePropertyType">
  <xs:attribute
    name="qName"
    type="xs:QName"
    use="required" />
  <xs:attribute
    name="populateFromFile"
    type="xs:boolean"
    use="required">
    <xs:annotation>
      <xs:documentation>
        Indicates whether the value of the metadata should be loaded
        from a file or will be populated by the implementation.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute
    name="fileLocation"
    type="xs:string"
    use="optional">
    <xs:annotation>
      <xs:documentation>
        If populateFromFile is true this will be the relative path to
        the file name of the file containing the instance data for
        this particular resource property. NOTE: this path is relative
        path from the etc directory of the service and should be
        located somewhere inside that etc directory.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute
    name="register"
    type="xs:boolean"
    use="required">
    <xs:annotation>
      <xs:documentation>
        Indicates whether this metadata should be registered if the
        service performs registration.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute
    name="description"
```

```
      type="xs:string"
      use="optional" />
  </xs:complexType>
</xs:schema>
```

gme://gov.nih.nci.cagrid.introduce/1/Extension

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Extension"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Extension"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element
    name="Extensions"
    type="ExtensionsType" />
  <xs:complexType name="ExtensionsType">
    <xs:sequence>
      <xs:element
        ref="Extension"
        minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element
    name="Extension"
    type="ExtensionType" />
  <xs:element name="ExtensionDescription">
    <xs:annotation>
      <xs:documentation>
        he extension type states whether or not this is a discovery type
        or service type extension. The extension preferencesPanel is a
        java classname which extends the
        ExtensionPreferencesConfigurationPanel
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="DiscoveryExtensionDescription" />
          <xs:element ref="ServiceExtensionDescription" />
          <xs:element ref="ResourcePropertyEditorExtensionDescription" />
          <xs:element ref="AuthorizationExtensionDescription" />
        </xs:choice>
        <xs:element
          ref="UpgradesDescription"
          minOccurs="0"
          maxOccurs="1" />
      </xs:sequence>
      <xs:attribute
        name="version"
        type="xs:string" />
      <xs:attribute
        name="extensionType"
        type="xs:string" />
```

```
    <xs:attribute
      name="extensionPreferencesPanel"
      type="xs:string" />
  </xs:complexType>
</xs:element>
<xs:element
  name="UpgradesDescription"
  type="UpgradesDescriptionType" />
<xs:complexType name="UpgradesDescriptionType">
  <xs:sequence>
    <xs:element
      ref="UpgradeDescription"
      minOccurs="0"
      maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:element
  name="UpgradeDescription"
  type="UpgradeDescriptionType" />
<xs:complexType name="UpgradeDescriptionType">
  <xs:attribute
    name="fromVersion"
    type="xs:string"
    use="optional" />
  <xs:attribute
    name="toVersion"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="upgradeClass"
    type="xs:string"
    use="required" />
</xs:complexType>
<xs:element
  name="ResourcePropertyEditorExtensionDescription"
  type="ResourcePropertyEditorExtensionDescriptionType" />
<xs:complexType
  name="ResourcePropertyEditorExtensionDescriptionType">
  <xs:sequence>
    <xs:element
      name="ResourcePropertyEditorPanel"
      type="xs:string"
      minOccurs="1"
      maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          Must extend the ResourcePropertyEditorPanel
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="Properties" />
  </xs:sequence>
  <xs:attribute
    name="qname"
    type="xs:QName"
    use="required" />
  <xs:attribute
```

```
    name="displayName"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="name"
    type="xs:string"
    use="required" />
</xs:complexType>
<xs:element
  name="DiscoveryExtensionDescription"
  type="DiscoveryExtensionDescriptionType" />
<xs:complexType name="DiscoveryExtensionDescriptionType">
  <xs:sequence>
    <xs:element
      name="DiscoveryToolsPanelExtension"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          Must extend the NamespaceTypeToolsComponent
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element
      name="DiscoveryPanelExtension"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
        <xs:documentation>
          Must extend the NamespaceTypeDiscoveryComponent
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="Properties" />
  </xs:sequence>
  <xs:attribute
    name="displayName"
    type="xs:string"
    use="required" />
  <xs:attribute
    name="name"
    type="xs:string"
    use="required" />
</xs:complexType>
<xs:element
  name="ServiceExtensionDescription"
  type="ServiceExtensionDescriptionType" />
<xs:complexType name="ServiceExtensionDescriptionType">
  <xs:sequence>
    <xs:element
      name="CreationPostProcessor"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
```

```
    <xs:documentation>
      Must implement the CreationExtensionPostProcessor
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element
  name="CreationUIDialog"
  type="xs:string"
  minOccurs="0"
  maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      GUI component that must extend a CreationExtensionUIDialog
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element
  name="CodegenPreProcessor"
  type="xs:string"
  minOccurs="0"
  maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      implement the CodegenExtensionPreProcessor
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element
  name="CodegenPostProcessor"
  type="xs:string"
  minOccurs="0"
  maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      implement the CodegenExtensionPostProcessor
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element
  name="ServiceModificationUIPanel"
  type="xs:string"
  minOccurs="0"
  maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      ModificationViewer GUI component that must extend a
      ServiceModificatoinUIPanel
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element
  name="ServiceDeploymentUIPanel"
  type="xs:string"
  minOccurs="0"
  maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
```

```
            Deployment GUI component that must extend a
            ServiceDeploymentUIPanel. This must not edit anything
            introduce managed such as the model. Only to be used for
            configureation etc.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element ref="Properties" />
    </xs:sequence>
    <xs:attribute
      name="displayName"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="name"
      type="xs:string"
      use="required" />
  </xs:complexType>
  <xs:element
    name="AuthorizationExtensionDescription"
    type="AuthorizationExtensionDescriptionType" />
  <xs:complexType name="AuthorizationExtensionDescriptionType">
    <xs:sequence>
      <xs:element
        name="ServiceAuthorizationPanel"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
      <xs:element
        name="MethodAuthorizationPanel"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
      <xs:element ref="Properties" />
    </xs:sequence>
    <xs:attribute
      name="displayName"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="name"
      type="xs:string"
      use="required" />
  </xs:complexType>
  <xs:element name="Properties">
    <xs:annotation>
      <xs:documentation>
        To be used for extension configuration properties
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Property"
          minOccurs="0"
          maxOccurs="unbounded">
          <xs:complexType>
```

```
            <xs:attribute
              name="makeGlobal"
              type="xs:boolean"
              use="optional" />
            <xs:attribute
              name="type"
              type="PropertyTypes"
              use="optional" />
            <xs:attribute
              name="key"
              type="xs:string"
              use="required" />
            <xs:attribute
              name="value"
              type="xs:string"
              use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ExtensionType">
    <xs:sequence>
      <xs:element
        name="ExtensionData"
        minOccurs="0"
        maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:any
              processContents="skip"
              minOccurs="0"
              maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute
      name="name"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="extensionType"
      type="xs:string" />
    <xs:attribute
      name="version"
      type="xs:string"
      use="optional" />
  </xs:complexType>
  <xs:simpleType name="PropertyTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="SERVICE_URL" />
      <xs:enumeration value="GENERAL" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

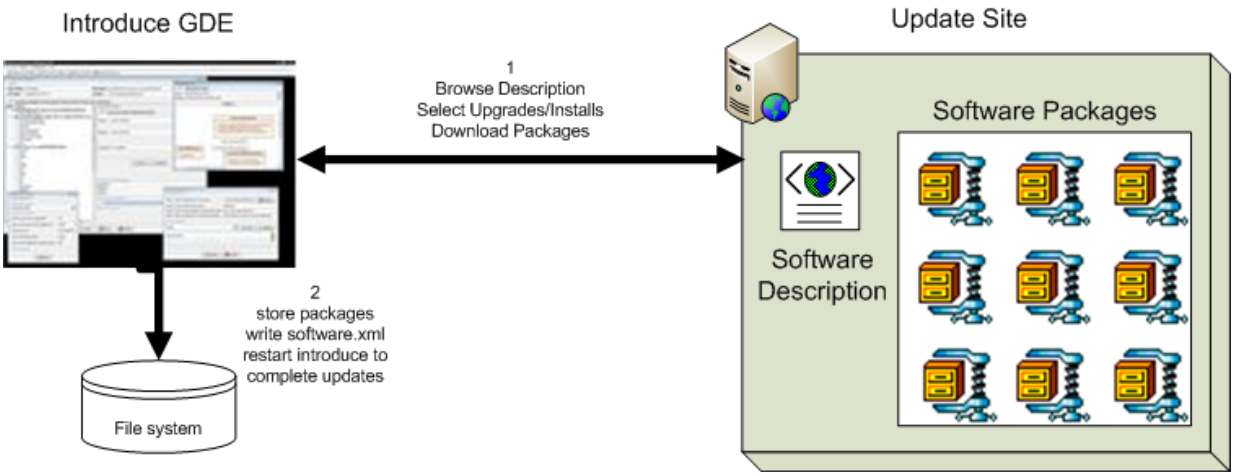
gme://gov.nih.nci.cagrid.introduce/1/Software

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns="gme://gov.nih.nci.cagrid.introduce/1/Software"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="gme://gov.nih.nci.cagrid.introduce/1/Software"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element
    name="Software"
    type="SoftwareType" />
  <xs:complexType name="SoftwareType">
    <xs:sequence>
      <xs:element
        ref="Introduce"
        minOccurs="0"
        maxOccurs="unbounded" />
      <xs:element
        ref="Extension"
        minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element
    name="Introduce"
    type="IntroduceType">
  </xs:element>
  <xs:complexType name="IntroduceType">
    <xs:attribute
      name="version"
      use="required"
      type="xs:string">
    </xs:attribute>
    <xs:attribute
      name="zipFileURL"
      use="required"
      type="xs:anyURI">
    </xs:attribute>
  </xs:complexType>
  <xs:element
    name="Extension"
    type="ExtensionType">
  </xs:element>
  <xs:complexType name="ExtensionType">
    <xs:attribute
      name="name"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="displayName"
      type="xs:string"
      use="required" />
    <xs:attribute
      name="version"
      type="xs:string"
```

```
      use="required">
    </xs:attribute>
    <xs:attribute
      name="compatibleIntroduceVersions"
      type="xs:string"
      use="required">
    </xs:attribute>
    <xs:attribute
      name="zipFileURL"
      use="required"
      type="xs:anyURI">
    </xs:attribute>
  </xs:complexType>
</xs:schema>
```

Introduce Update Framework

The introduce update framework enables the introduce software systems and its extensions to be able to update themselves with new versions of software from designates update sites. This will enable the software to be able to freely move forward on customer machines as new features are available. The basic flow is that Introduce will enable upgrage packages to be downloaded from an update site. Once then packages are downloaded the Introduce GDE will be restarted. On exit, Introduce will examine the *updates* directory to determine if there are any packages to be installed. If there are Introduce will unzip them in the appropriate place depending on whether the package is an Introduce package or an Extension package. Once these extractions are complete for each update, Introduce will clean out the *updates* directory and re-start the Introduce GDE. The image below shows in better detail the interactions that will happen. For more information on standing up an update site refer to the Introduce Developers Guide.



Introduce Service Migration Framework

The Service Migration Framework provides support for migrating services build with older versions on Introduce and Introduce Extensions to newer versions. The Framework is broken down into 4 main peices. The Upgrade Manager which is responsible for orchestrating the upgrading of the service. The Model Upgrader is responsible for migrating the services introduce.xml file to the new model of the file if there are changes to the Introduce Model. The Introduce Upgrader will upgrade make any changes to the services that Introduce is responsible for such as libraries, build files, source code, etc. And lastly, the Extension Upgraders which are responsible for upgrading any pieces of the service which the particular extension is responsible for editing.

- Prerequisite
  - Older versions of the introduce model will be packaged in a new java package and will be passed into

older version upgraders so that they can deserialize and use the older version of the model.

- The Upgrade Cycle
  1. 1. run the *model upgrader* to bring the model to the next minor version of introduce version so that the model can be used
  2. 2. run the *introduce upgrader* for that next version so that introduce managed service files can be brought up to the intermediate version of introduce
     - for each minor version increment call any available *extension upgraders* that can upgrade an extension for the current version of introduce. The current versions model will be passed into the extension upgrader.
     - rinse and repeat.

Introduce Upgrader Base

In order to provide an upgrader from one version of an Introduce to another a developer would have to extend this class and implement the *upgrade* operation.

```
package gov.nih.nci.cagrid.introduce.upgrade.one.one;
```

```
import gov.nih.nci.cagrid.introduce.common.ServiceInformation;
import gov.nih.nci.cagrid.introduce.upgrade.common.IntroduceUpgradeStatus;
import gov.nih.nci.cagrid.introduce.upgrade.common.IntroduceUpgraderI;
import gov.nih.nci.cagrid.introduce.upgrade.common.StatusBase;


public abstract class IntroduceUpgraderBase implements  IntroduceUpgraderI{
    ServiceInformation serviceInformation;
    IntroduceUpgradeStatus status;
    String fromVersion;
    String toVersion;
    String servicePath;


    public IntroduceUpgraderBase(IntroduceUpgradeStatus status,
ServiceInformation serviceInformation, String servicePath, String
fromVersion,
        String toVersion) {
        this.status = status;
        this.serviceInformation = serviceInformation;
        this.fromVersion = fromVersion;
        this.toVersion = toVersion;
        this.servicePath = servicePath;
        status.setFromVersion(fromVersion);
        status.setToVersion(toVersion);
        status.setType(StatusBase.UPGRADE_TYPE_INTRODUCE);
        status.setName("IntroduceUpgrader " + fromVersion + " - " +
toVersion);
    }


    public void execute() throws Exception {
        System.out.println("Upgrading Introduce Service From Version " +
this.getFromVersion() + " to Version "
            + this.getToVersion());
        upgrade();

getServiceInformation().getServiceDescriptor().setIntroduceVersion(getToVersi
on());
    }


    public String getFromVersion() {
        return fromVersion;
    }


    public void setFromVersion(String fromVersion) {
        this.fromVersion = fromVersion;
    }


    public String getToVersion() {
        return toVersion;
    }
```

```
    public void setToVersion(String toVersion) {
        this.toVersion = toVersion;
    }


    protected abstract void upgrade() throws Exception;


    public ServiceInformation getServiceInformation() {
        return serviceInformation;
    }


    public void setServiceInformation(ServiceInformation serviceInformation)
{
        this.serviceInformation = serviceInformation;
    }


    public String getServicePath() {
        return servicePath;
    }


    public void setServicePath(String servicePath) {
        this.servicePath = servicePath;
    }


    public IntroduceUpgradeStatus getStatus() {
        return status;
    }


    public void setStatus(IntroduceUpgradeStatus status) {
        this.status = status;
    }

}
```

Extension Upgrader Base

In order to provide an upgrader from one version of an Introduce Extension to another a developer would have to extend this class and implement the *upgrade* operation.

```
package gov.nih.nci.cagrid.introduce.upgrade.one.one;

import gov.nih.nci.cagrid.introduce.beans.extension.ExtensionType;
import gov.nih.nci.cagrid.introduce.common.ServiceInformation;
import gov.nih.nci.cagrid.introduce.upgrade.common.ExtensionUpgradeStatus;
import gov.nih.nci.cagrid.introduce.upgrade.common.ExtensionUpgraderI;

/**
```

```
 * Class must be extended to provide an extension upgrader. An extension
 * upgrader should be used to provide upgrades to the service to support a
newer
 * version of an extension. The extension upgrader should only touch parts of
 * the service that the extension itself controls, i.e. the extension data in
 * the xml entity in the introduce.xml document in the service's top level
 * directory. The version attribute in the extensionType will automatically
be
 * updated.
 *
 * @author hastings
 *
 */
public abstract class ExtensionUpgraderBase implements ExtensionUpgraderI {

      ExtensionType extensionType;
         ServiceInformation serviceInformation;
          String fromVersion;
          String toVersion;
          String servicePath;
          ExtensionUpgradeStatus status;

      public ExtensionUpgraderBase(String upgraderName, ExtensionType
extensionType,
                  ServiceInformation serviceInformation, String servicePath,
                  String fromVersion, String toVersion) {
          this.serviceInformation = serviceInformation;
        this.fromVersion = fromVersion;
        this.toVersion = toVersion;
        this.servicePath = servicePath;
            this.extensionType = extensionType;
            this.status = new
ExtensionUpgradeStatus(upgraderName,this.extensionType.getName(),this.fromVer
sion,this.toVersion);
      }

      public void execute() throws Exception {
            System.out.println("Upgrading services " +
extensionType.getName()
                        + " extension  from Version " + this.getFromVersion()
                        + " to Version " + this.getToVersion());
            upgrade();
            extensionType.setVersion(getToVersion());
      }

      public ExtensionUpgradeStatus getStatus(){
          return this.status;
      }

      public ExtensionType getExtensionType() {
            return extensionType;
      }

      public void setExtensionType(ExtensionType extensionType) {
            this.extensionType = extensionType;
      }
```

```
          public String getFromVersion() {
              return fromVersion;
           }

           public void setFromVersion(String fromVersion) {
              this.fromVersion = fromVersion;
           }

           public String getToVersion() {
              return toVersion;
           }

           public void setToVersion(String toVersion) {
              this.toVersion = toVersion;
           }

           protected abstract void upgrade() throws Exception;

           public ServiceInformation getServiceInformation() {
              return serviceInformation;
           }

           public void setServiceInformation(ServiceInformation
serviceInformation) {
              this.serviceInformation = serviceInformation;
           }

           public String getServicePath() {
              return servicePath;
           }

           public void setServicePath(String servicePath) {
              this.servicePath = servicePath;
           }
}
```