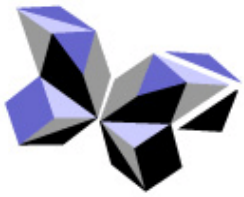


Digital Business Ecosystem

Contract n°507953

Workpackage WP26: DBE Portal

Deliverable D26.6: DBE Portal Specification

Information Society Technologies

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°: D26.6

Due dates: 12/2005

Delivery Date: 02/2006

Short Description:

A Digital Business Ecosystem Portal will be a user friendly entry point to the DBE that provides the means to search, browse and execute DBE Services over the DBE Peer-To-Peer network using nothing more than a web browser. This document sets out the design of the DBE Portal and the requirements to satisfy this design. It also details the implementation of the supporting DBE Portal toolkit and any issues encountered in its implementation.

Author: Intel Ireland Ltd.

Partners contributed:

Made available to: Public

Versioning

Version	Date	Author, Organisation
0.1	24/05/2005	Andy Edmonds, TCD. Initial Draft
0.2	28/11/2005	Andy Edmonds, Intel Ireland Ltd. Major revision to original initial draft.
0.3	17/02/2005	Andy Edmonds, Intel Ireland. Final modifications and correction before submittal.
0.4	28/02/2006	Andy Edmonds, Intel Ireland Ltd. Suggestions from internal reviewer added.
0.5	09/03/2006	Andy Edmonds, Intel Ireland Ltd. Further suggestions from internal reviewers added.

Quality check:

1st Internal Reviewer : Pierfranco Ferronato, Soluta.

2nd Internal Reviewer: Juanjo Aparicio, Techideas.

Table of Contents

Index of Figures

Index of Tables

1 Introduction

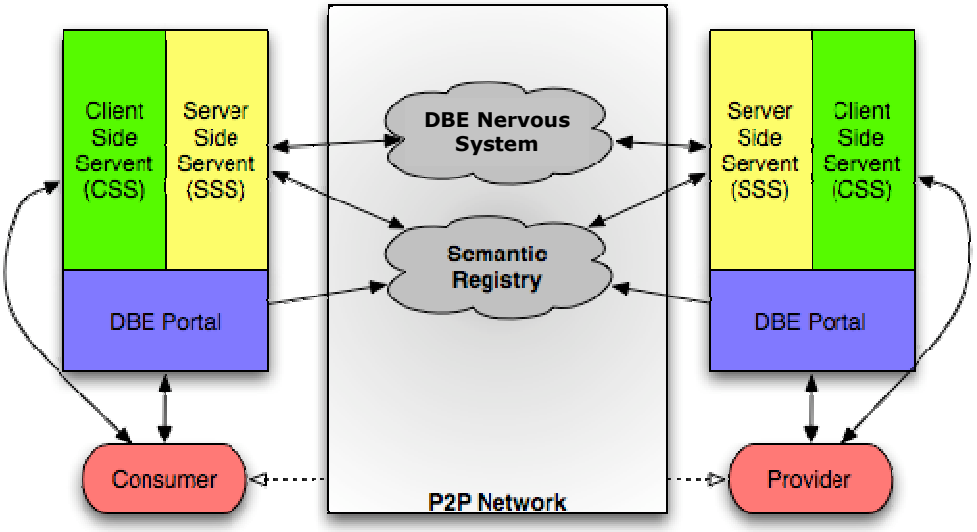


Figure 1 An Overview of the DBE Portal

A Digital Business Ecosystem (DBE) Portal is a user friendly entry point to the DBE that provides the means to search, browse and execute DBE Services over the DBE Peer-To-Peer (P2P) network using nothing more than a web browser. Its relationship with the Execution Environment (ExE) is shown in Figure 1. A DBE Portal empowers any user of the DBE with a minimum of technical know-how, and without any installation of DBE applications, to access, consume and utilise the DBE and the services it hosts.

A DBE Portal is an evolved and enhanced DBE Desktop [1]. The initial release of the DBE Portal seeks to match the functionality already contained in the DBE Desktop. The main functionality presented in the DBE Portal, just as in the DBE Desktop, is the ability to search, browse and execute services. The development of the DBE Desktop has halted except for critical bug fixes. Future feature additions and functionality enhancements will be folded into the DBE Portal development effort.

The method of interfacing with the DBE Portal is by using any web browser (e.g. Firefox, Internet Explorer, Mozilla, and Opera). The DBE Portal does not require any heavy client side plug-ins such as Java run-time environments and as such the web browser is the thin client. All computation is performed on the servent, through the DBE Portal, with the results of computation

displayed on the client (web browser). The computation on the servent [8] is performed by interfacing with the DBE via the toolkits and frameworks provided by the Java development kit (JDK) and those already developed by DBE partners. These include the servent Application Programming Interfaces (API) and the Abstract Protocol Adaptor (APA). The framework that allows DBE services to display user interfaces within the client (web browser) is specified in deliverable 20.1 [2]. A programmer's toolkit, the Portal Toolkit, has been developed to allow the execution of services through user interfaces (UI) displayed by the end-users' web browsers.

Taking this approach allows any user of the DBE, without an installation of any DBE applications, to access the DBE with just the bare minimum of a web browser.

The DBE Portal is dependent on other DBE components as shown in the diagram below (Figure 2). Where a user does not have the facility to run their own servent and consequently a DBE Portal, it could be possible, where a Small-to-Medium Enterprise (SME) allows it, for that user to use another SME's DBE Portal to search, browse and execute DBE services. For this to be an attractive proposition for a SME to consider, incentives must be offered to encourage them to allow lone users access their DBE Portal.

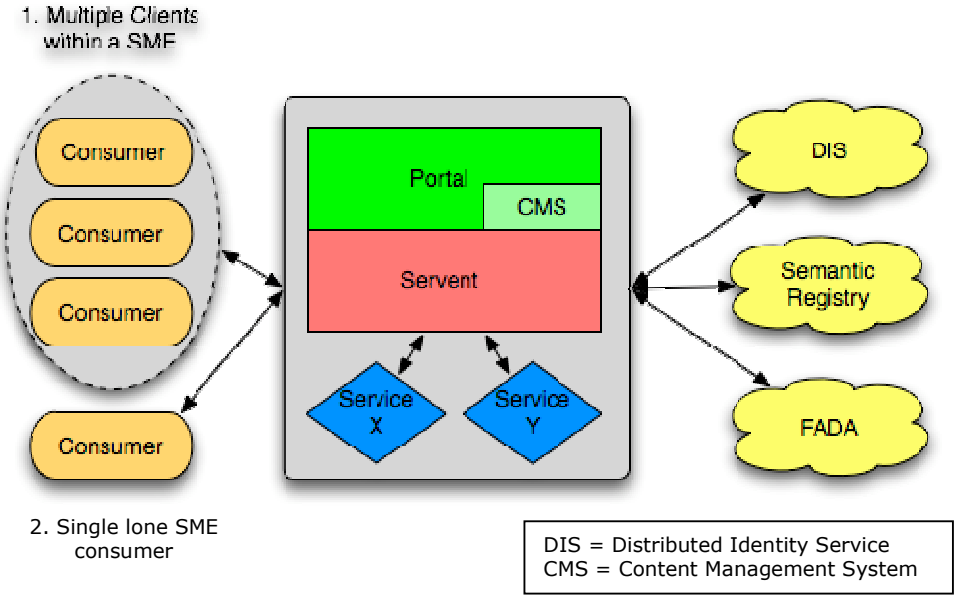


Figure 2 DBE Portal and Related Components

2 DBE Portal Feature Requirements

The basic premise of the DBE Portal is to allow any user of the DBE, be they consumers[1] or providers[2], to interact with services advertised in the DBE using the following three basic functionalities:

- *Search* – using the DBE Portal a DBE user can provide search terms that describe the service that the user is looking for. The returned results are services that best match the entered search terms.

- *Browse Service Information* – After submitting a search request and retrieving back the results, the user can view additional information about a service that the user wants to use.

- *Execute* – using the results from the action of *searching* or *browsing*, once a suitable service is selected, the user can invoke and execute that service. From the point of view of the DBE Portal, the execution of a service involves the retrieval of the user interface of that service, and displaying it to the requesting user. On entering data, the user can then *execute* the UI's functionality.

It is this set of functionality that has been implemented in the first version of the DBE Portal. The above features fall into the category of application/service functionality and are features related to direct manipulation of DBE services. With the necessary features (listed above) implemented, it is envisioned that new and extra functionalities will be added to the DBE Portal especially those encapsulated by the general notion of account management, which includes tasks such as service bookmarking, profile management, workflow management etc.

Although the DBE Portal has been designed to satisfy these basic and mandatory use cases, additional functionality has also been considered. Although considered important, it should be noted that the additional functionality will only be implemented when the core functionality of the DBE Portal is complete and released. The secondary functionalities (use cases) fall into the categories of infrastructural or informational and include:

[1] A consumer can be viewed as a classic client; an entity that uses the functionality of a service to achieve an end-goal.
[2] A provider, in the sense of DBE, is an entity that provides a service that can be consumed. The service that the provider offers may utilise external services and in this provider specific context, the provider is also a consumer too.

- Facilities to provide a free “web-presence” of the SME not only to users within the DBE network but also to the wider population of users that use the World Wide Web (WWW). This will provide a means for the SME to disseminate information about itself and its services. To this end, a Content Management System (CMS) for the organisation of SME information may be used. A CMS is a system used to organize and facilitate collaborative content creation. It provides a simple way to create, maintain and update content hosted on a DBE Portal and display it using the familiar mechanisms of a web server via a “web site”.

- Using a DBE Portal, a DBE user could, starting from a particular taxonomy, drill down until a service meeting the user’s requirements is discovered. The desired functionality of browsing DBE service could emulate that of Universal Description, Discovery and Integration (UDDI) service listing, for example:
    - White directory: based on address and contact information. This information can be found within a service’s Business Modelling Language (BML) data model (M0), contained in the service’s service manifest.
    - Yellow directory: based on service categorised by industry/business. This information can be found within a service’s BML model (M1), contained in the service’s service manifest
    - Green directory: based on technical information about services. This type of information can be found within a service’s Service Definition Language (SDL) model, contained in the service’s service manifest.

- A means of user registration. This will allow registration of new identities with the DBE using the Distributed Identity Service (DIS) as it is anticipated that DBE Portals will be entry points (i.e. registration agents) for an SME wishing to join the DBE. It will be here where the SME will register with the DBE in order to provide its services.

- A way to allow user profile creation and management. This should facilitate the modification and management of a registered user’s profile [3] identified by the user’s DIS identity. Using the user’s profile, it is hoped that preferred services can be bookmarked and stored by saving a reference to the service’s Service Manifest Identifier (SMID). It is foreseen that the Distributed Storage System (DSS) could be used as a mechanism to allow the storage of such information and the user’s profile.

- A DBE Portal should also be registered with a directory-type component (see Ch. 4) when the SME starts its servent and comes online. Conversely, it should be unregistered when the SMEs servent is shutdown and goes offline. This task may also involve updating of the DBE Portal's endpoint information with the directory-type component especially if the SME's Internet Service Provider (ISP) mandates the use of Dynamic Host Control Protocol (DHCP) [10] allocated Internet Protocol (IP) [11] addresses and or the DBE Portal is located behind a Network Address Translator (NAT) [12].

- A distributed DBE service development environment where DBE developers can create service compositions and deploy those compositions. User interfaces could, potentially, also be created within a DBE Portal to visually represent those service compositions to consumers.

- A means and mechanism to access the DBE applications. The DBE suite of applications enables the development of DBE services, the consumption of services and the provisioning of DBE services. These applications include the DBE Studio and related plugins, the DBE Servent and the DBE Evolutionary Environment (EvE). Also any frameworks that a developer may be interested in using e.g. the APA should be accessible.

- A place where DBE related statistics can be viewed, e.g., number of Knowledge Base (KB) nodes, number of currently "live" SMEs, total number of "live" DBE services, network topology and it size. P2P system / KB can't give us absolute data. This task could be performed by another DBE core service that DBE Portal could utilise, which could perhaps supply greater information than the partial view of the network available to the DBE Portal. This service could track various P2P system nodes and topologies formed between servent and P2P system nodes. Another, but simpler way to get some statistics would be to issue a query to the Semantic Registry (SR) requesting how many entities of the DBE Portal type are currently registered.

- Support for the resumption of DBE user sessions could also be included in the portal so that DBE users could resume work from where they last left after logging out of DBE. By supporting this, it would enable the retrieval of results from long running transactions as it is infeasible to expect DBE users to remain logged in to the DBE over period of days. The user's state could be saved in DSS.

3 DBE Portal Use Cases

What follows is a listing of possible use cases related to the DBE Portal. Those listed below may not be implemented in the initial release of the DBE Portal but will start to form a development road map for the DBE Portal. From the DBE technical annex, it is stated that there will be two releases, both deliverables, of the DBE Portal corresponding to the versions of DBE Portal Version 1 and DBE Portal Version 2. Below is a table summarising the use cases collected so far for the DBE Portal.

Use Case Reference	Use Case Description
UUC1	Find Services Use-case
UUC2	Browse Service Information Use-case
UUC3	Invoke Service UI Use-case
UUC4	Manage Profile Use-case
UUC5	Register with Identity Service Use-case
UUC6	Login/Restore Session Use-case
UUC7	Access DBE Components Use-case
UUC8	Browse DBE Services Use-case
PoUC1	Disseminate SME Information Use-case
PoUC2	Register With DBE Use-case
PrUC1	Author Service UI Use-case
PrUC2	Compose Service Use-case
PrUC3	Administer Service Use-case
PrUC4	Deploy Service Use-case
CUC1	Bookmark Service Use-case

Table 1 Reference Table of Use Cases

3.1 DBE User Use-cases

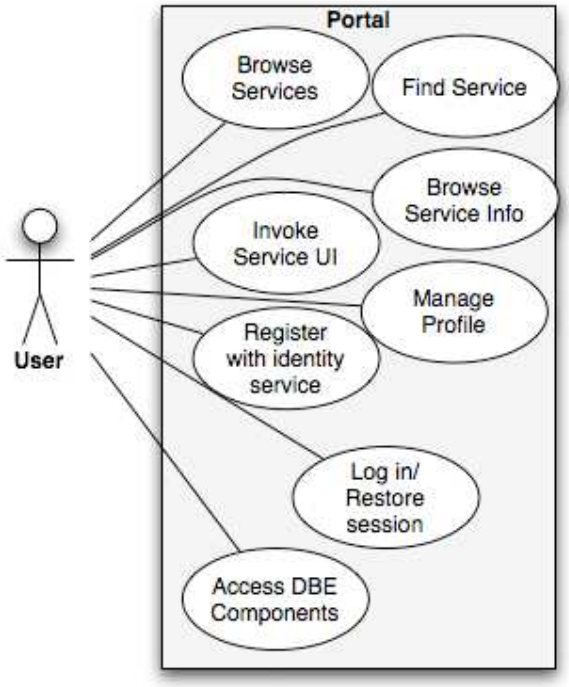


Figure 3 DBE User Use-cases

Use Case UUC1	**Find Service**
Area	DBE Portal
Objective	Find existing DBE Service through a Web Browser
Actors	User
Pre-Conditions	DBE Services must be available
Post-Conditions	
Description	The user will issue search queries in a Web Browser in order to find services he/she is interested in.
Dependencies	User Interface (web-based), Semantic Registry
Tentative Release	Version 1
Relevant Partner	TUC, SUN, Intel

Table 2 Find Services Use-case

Use Case UUC2	**Browse Service Information**
Area	DBE Portal
Objective	Browse DBE Service Information through a Web Browser
Actors	User
Pre-Conditions	DBE Services must be available
Post-Conditions	
Dependencies	Semantic Registry, Portal Toolkit
Tentative Release	Version 1
Relevant Partner	TUC, Intel

Table 3 Browse Service Information Use-case

Use Case UUC3	**Invoke Service UI**
Area	DBE Portal
Objective	Utilise an existing DBE Service's UI through a Web Browser
Actors	User
Pre-Conditions	DBE Services must be available The search query must return at least two DBE Services
Post-Conditions	
Description	The user will utilise an existing DBE Service through a Web Browser. The service will display an Openlaszlo user interface with input fields to fill in by the user.
Dependencies	Web-based GUI Development Tool, Portal Toolkit
Tentative Release	Version 1
Relevant Partner	Intel, SUN

Table 4 Invoke Service UI Use-case

Use Case UUC4	**Manage Profile**
Area	DBE Portal
Objective	Edit, update and manage a user's DBE preferences.
Actors	User
Pre-Conditions	User is logged in. User-profile service is available
Post-Conditions	
Description	The user can manage his/her preferences and persist these to suitable storage
Dependencies	User profiling service
Tentative Release	Version 2
Relevant Partner	FZI, Intel

Table 5 Manage Profile Use-case

Use Case UUC5	**Register with Identity Service**
Area	DBE Portal
Objective	Issue a registration request for a DBE Identity and receive corresponding credentials
Actors	User
Pre-Conditions	User is not registered with the identity service
Post-Conditions	User has a DBE identity
Description	A user can register for a DBE Identity though the DBE Portal
Dependencies	Identity service.
Tentative Release	Version 2
Relevant Partner	TCD, Intel

Table 6 Register with Identity Service Use-case

Use Case UUC6	**Login/Restore Session**
Area	DBE Portal
Objective	Return to a long-running session through the DBE Portal
Actors	User
Pre-Conditions	An existing session running
Post-Conditions	

Description	When a user executes a long-running process e.g. a workflow, the user will be able to log out and at a later stage log back in and check the status of the process
Dependencies	Identity service, user-profiling service
Tentative Release	Version 2
Relevant Partner	University of Surrey, Intel

Table 7 Login/Restore Session Use-case

Use Case UUC7	**Access DBE Components**
Area	DBE Portal
Objective	Download DBE components to develop and access DBE services
Actors	User
Pre-Conditions	
Post-Conditions	
Description	A user wanting to develop DBE services can access the necessary resources to accomplish this.
Dependencies	Sourceforge project site
Tentative Release	Version 1
Relevant Partner	Intel

Table 8 Access DBE Components Use-case

Use Case UUC8	**Browse DBE Services**
Area	DBE Portal
Objective	Browse DBE Service Information through a Web Browser in a UDDI fashion
Actors	User
Pre-Conditions	DBE Services must be available
Post-Conditions	
Description	
Dependencies	Semantic Registry, Portal Toolkit
Tentative Release	Version 2
Relevant Partner	TUC, Intel

Table 9 Browse DBE Services Use-case

3.2 DBE Portal Use-cases

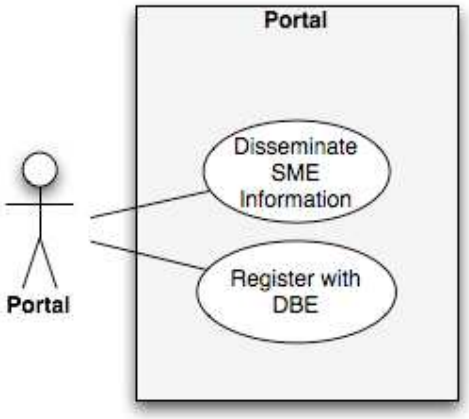


Figure 4 DBE Portal Use-cases

Use Case PoUC1	**Disseminate SME Information**
Area	DBE Portal
Objective	Provide an "out of the box" web site.
Actors	Portal
Pre-Conditions	
Post-Conditions	
Description	This will provide a free web presence that will allow SMEs provide more information about its business.
Dependencies	Content Management System (Possibly)
Tentative Release	Version 1
Relevant Partner	Intel

Table 10 Disseminate SME Information Use-case

Use Case PoUC2	**Register With DBE**
Area	DBE Portal
Objective	Register the Portal with DBE infrastructure
Actors	Portal
Pre-Conditions	
Post-Conditions	
Description	By registering the Portal with DBE infrastructure, it will be possible to search for Portals within a particular business domain.
Dependencies	KB, SR
Tentative Release	Version 2
Relevant Partner	Intel, SUN

Table 11 Register With DBE Use-case

3.3 DBE Provider Use-cases

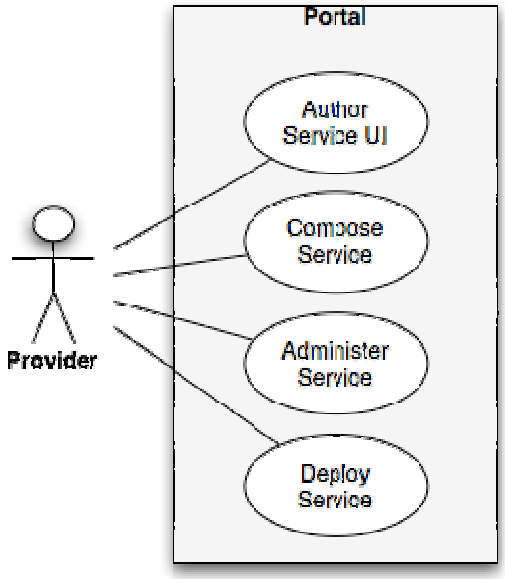


Figure 5 DBE Provider Use-case

Use Case PrUC1	**Author Service UI**
Area	DBE Portal
Objective	Allow users create declarative UIs within the Portal
Actors	Provider
Pre-Conditions	
Post-Conditions	
Description	A user can use the Portal to author declarative UIs and associating it with a service on the portal's servent
Dependencies	UI Technology
Tentative Release	Version 2
Relevant Partner	SUN, Intel

Table 12 Author Service UI Use-case

Use Case PrUC2	**Compose Service**
Area	Portal
Objective	Allows users to create service compositions
Actors	Provider
Pre-Conditions	
Post-Conditions	
Description	A user will be able to create service compositions of services found within the DBE within the Portal
Dependencies	Service composition engine
Tentative Release	Version 2
Relevant Partner	TCD, Intel

Table 13 Compose Service Use-case

Use Case PrUC3	Administer Services
Area	Portal
Objective	Administer various services of the ExE
Actors	Provider
Pre-Conditions	
Post-Conditions	
Description	This will allow the administration of the various components of the ExE through the portal
Dependencies	P2P system, Servent, KB, SR
Tentative Release	Version 2
Relevant Partner	Intel, SUN

Table 14 Administer Service Use-case

Use Case PrUC4	Deploy Service
Area	Portal
Objective	Deploy an implemented service
Actors	Provider
Pre-Conditions	
Post-Conditions	
Description	This will allow the deployment of an implemented service through the Portal
Dependencies	Servent
Tentative Release	Version 2
Relevant Partner	Intel, SUN

Table 15 Deploy Service Use-case

3.4 DBE Consumer Use-cases

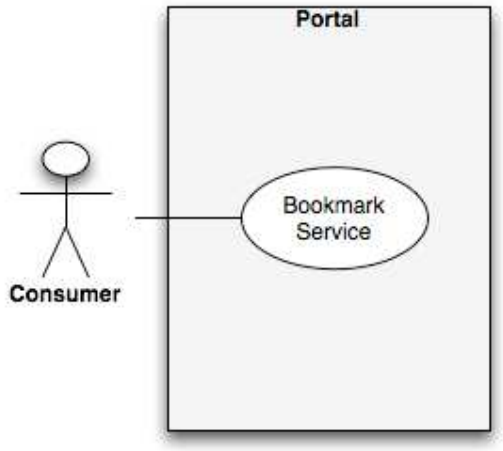


Figure 6 DBE Consumer Use-case

Use Case CUC1	**Bookmark Service**
Area	DBE Portal
Objective	Store pointers to used DBE service instances
Actors	Consumer
Pre-Conditions	Existing services
Post-Conditions	
Description	This will allow users store pointers to used DBE services much like a user would bookmark websites
Dependencies	
Tentative Release	Version 2
Relevant Partner	Intel, TUC

Table 16 Bookmark Service Use-case

4 DBE Portal Considerations

4.1 DBE Portal Registration

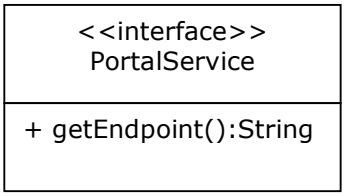
Originally in the initial stages of the design and specification of the DBE Portal, it was conceived that there would be a DBE Directory, a type of meta-portal named as the DBE Gateway Portal. It would be here where all DBE Portals hosted on servents would register themselves along with their endpoint information. It would be a directory that aggregated all SMEs currently online and organised them according to their ontology.

Although this approach would suffice for searching for DBE Portals and the services hosted within them, it would be unfortunately deficient. The primary reason to reject such a design is that in essence the DBE Gateway Portal would be a single point of failure in a supposed and otherwise fully decentralised system. Another major point of rejection is that parallel tracks of development would be necessitated to develop such a Gateway Portal. From exploratory work carried out by the author, it was viewed that a new lightweight directory service would be required to enable the Gateway Portal within DBE. Obviously this would circumvent established frameworks and core services already available within the DBE ecosystem and from a managerial point of view with regards to resource allocation would be inefficient and wasteful. Posed with these problems, it was necessary to find a solution using the current frameworks and DBE core services so DBE Portals could register themselves within the DBE networks so to be searchable.

To this end, the notion of modelling a DBE Portal as a DBE service was conceived. What is required for this solution is to describe the DBE Portal as a BML model, a model that should not change from SME to SME. Such a BML model is a model template. The BML template provides a means to search for a group of DBE Portals that fall under a particular type of business domain in a distributed P2P fashion. All that will change is the BML data associated with that model. The BML data required for a DBE Portal will be supplied when it is installed along with the servent and will be a one-time process.

By utilising features of BML, it is possible to organise SME DBE Portals into categories much in the same way as organisation is performed in folksonomies [4] by using the data modelled in the BML. By following such an approach it is feasible to imagine self-organising structure and categorisation within the DBE where more accurate terms describing a particular business domain are popularised. Those popularised are the terms that are the “fittest” for describing such a domain. It is important to note that by following this approach it does not require that an administrator look after a particular ontology and it is possible to organise services in a navigable, automatically created structure.

Although, initially it would make sense to store endpoint information in the BML data, this after further thought would be infeasible. It would only work where it can be guaranteed that for all SMEs their endpoint information was immutable and never changing, but for many SMEs this is not the case (e.g. where there IP address is allocated by DHCP). As BML data is considered to be slowly changing over time, it is not a suitable place to store endpoint information, which could be potentially updated every hour. To resolve this, a simple SDL model of a DBE Portal is required. It will be this SDL model and its corresponding interface and methods that will return back endpoint information to locate a SME's DBE Portal. The interface of the SDL model, shown as a Java interface for brevity, is envisioned as follows:



Snippet 1 DBE Portal SDL Interface

With the DBE Portal models created and implemented the DBE Portal service would be exported to the servent where it will be published (with its service manifest) to the Semantic Registry (SR) and have its service proxy registered with the P2P network. With this performed the DBE Portals contained within the DBE network are then searchable using the existing DBE infrastructure.

Typically, when a SME first wants to log onto the DBE, the SME starts the servent which in turn starts the DBE Portal. As the servent starts, it registers all services that are currently deployed on it including the DBE Portal service. As it registers the DBE Portal service, the servent should also update the DBE Portal's endpoint information by checking for:

- Renewed DHCP IP address – whether or not the SMEs ISP has requested that the SME's machine through which it accesses the network should renew its IP address.
- NAT Status – whether or not the servent is behind a NAT and if so its type and the external IP address the servent is contactable at. This NAT information could possibly be obtained by using a STUN [5] server.

Once the DBE Portal service has started up, the DBE Portal to which it corresponds is available through the established mechanisms of discovery and consumption in the DBE.

- *Discovery* is performed by the SR and consumption of the DBE Portal service is executed by the servent.
- *Consumption* happens when the DBE Portal service's proxy returns the endpoint of the DBE Portal where it can be located and displayed within a standard web browser.

When the SME shuts down the servent, its DBE Portal goes off-line and the service is shutdown thus removing it from the P2P system and making it unavailable. This validates the metaphor "the SME is closed for business" and although closed for business the SME still has a visibility within the DBE, just like a closed shop has on a street.

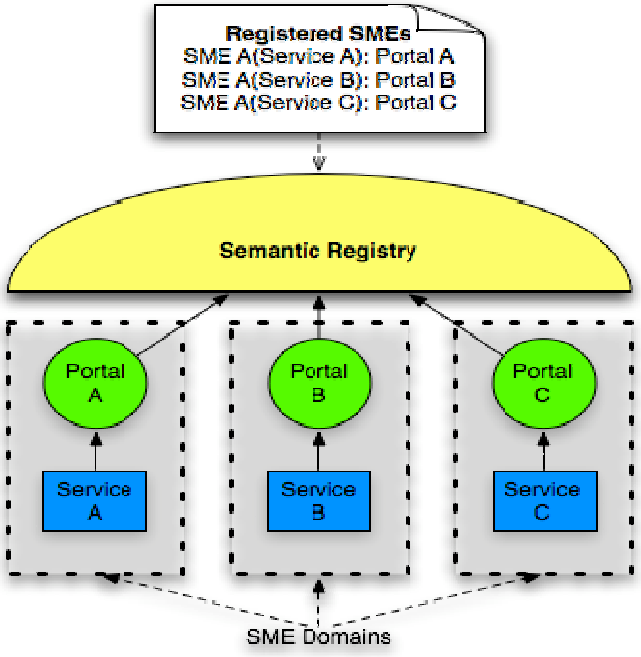


Figure 7 Registered DBE Portals and SMEs

4.2 Users and the DBE Portal

There are a number of scenarios in which users of a service can interact with the DBE Portal and its related services. The simplest case is that of a user, acting as a single entity, directly accessing a DBE Portal that is externally accessible and allows both anonymous and authenticated users interact with it. This is shown in Figure 8.

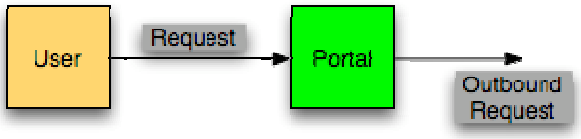


Figure 8 A Single User and a DBE Portal

A second case is where multiple users belonging to the same organisation or SME interact with their instance of a DBE Portal. In this case, users within the SME's domain are the only users interacting with it. This DBE Portal in this case is not accessible from domains outside of the SME's domain. This is shown in Figure 9.

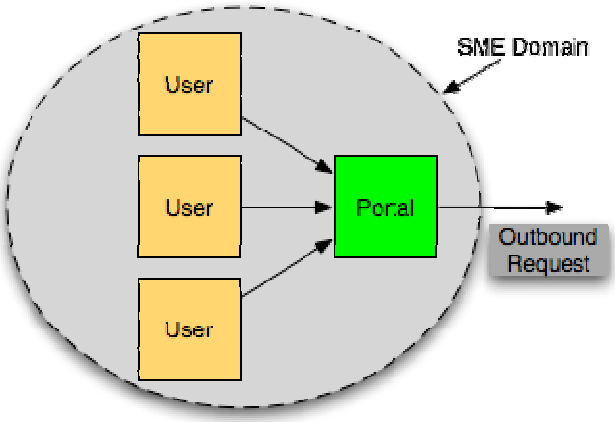


Figure 9 Multiple Users with the Same Domain and a DBE Portal

4.3 Security and the DBE Portal

This is related to the previous section on the topic of users and the DBE Portal. What is in question here is where and at what level is authentication and authorisation performed?

In the first case, the single user entity scenario, the single user remains anonymous, or authenticates against the DBE Portal. With the authentication information, it can then, on the behalf

of the user, authenticate against the Distributed Identity Service (DIS). If the DBE Portal allows for anonymous access, then the policies of what an anonymous user can do at that DBE Portal could be set by using the administration interface of it. Below, an example of a single user invoking a service with authentication is shown:

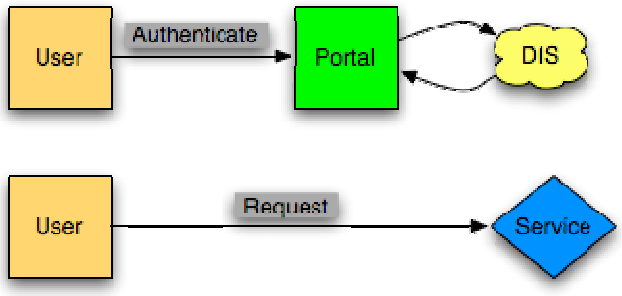


Figure 10 Single User Authentication

The next scenario is when there are multiple users within the same SME domain that require the use of services that reside outside of the SME's domain. The DBE Portal in this case is not accessible from domains outside of the SME's domain. There are two ways authentication can be performed in this scenario. The first method in which authentication can be performed is shown in Figure 11. In this scenario, the DBE Portal holds the SME's DBE credentials that allow users invoke services through it. It authenticates for the SME domain that it is part of. Any users within this domain can then use the services advertised within the DBE as it is assumed that they are part of the SME. Taking this approach allows for transparent authentication, makes the implementation of clients easier as they need not implement authentication mechanisms and allows the SME itself manage each of its users independently without having to go to a third party to modify a users permissions or access rights.

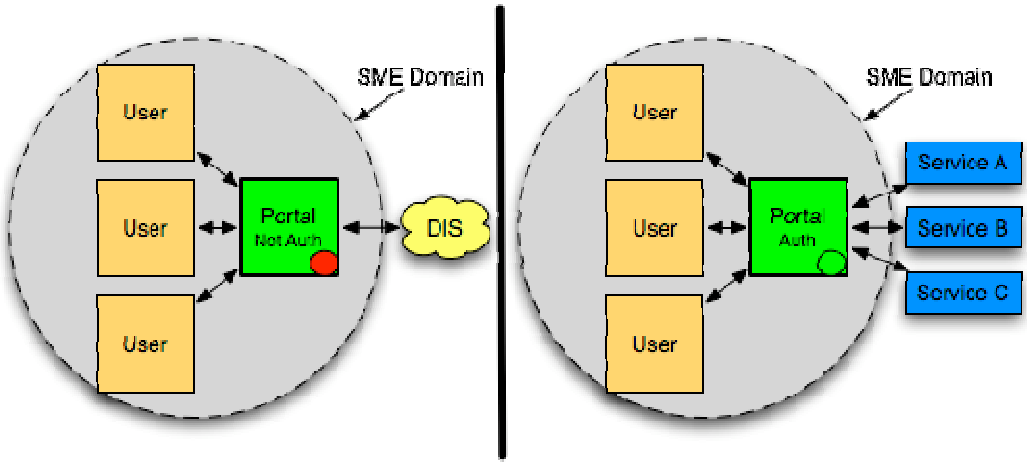


Figure 11 Multiple User Authentications: Scenario 1

The second way authentication can be performed with multiple users within a SME's domain is shown in Figure 12. In this scenario, every user within the SME's domain must authenticate against the SME's DBE Portal. It then authenticates on behalf of the user with the DIS. Once authenticated, a user then can use the services within the DBE. Using this scenario allows for each user within the SME to have a stronger identity as each user will have an identity within the DIS. This disables a masquerading attack where a hostile user initiates an attack of some sort. This attack would be possible if authentication is performed as in the previous scenario.

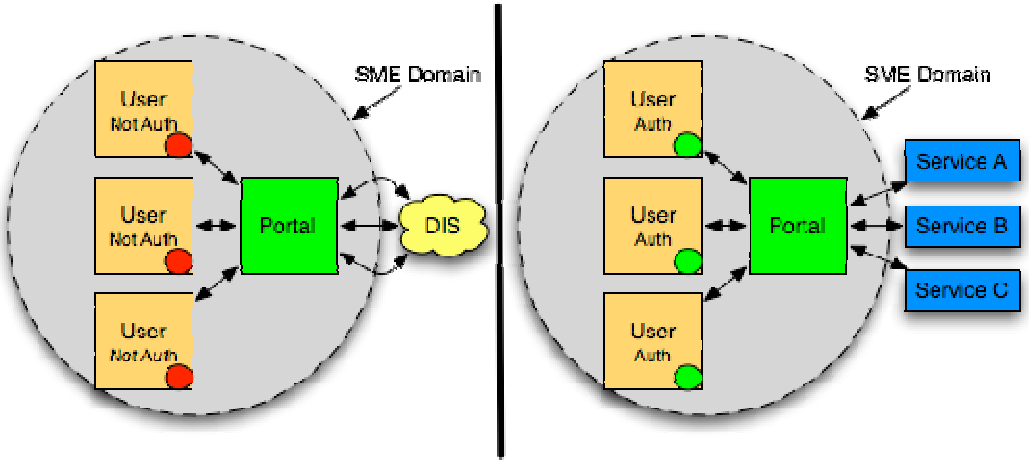


Figure 12 Multiple User Authentication: Scenario 2

5 DBE Portal Toolkit

The DBE Portal Toolkit is one that has been designed to support the:

- Retrieval of a remote UI on a remote servent and display of it using the local servent
- Asynchronous[3] remote execution of a service through a locally hosted user interface

Both the above tasks are carried out by the DBE Portal Toolkit that comprises of:

- DBE Portal Client – This component is responsible for the retrieval of UIs and execution of services represented by those UIs. This is a component available within the Openlaszlo engine, which runs on the Servent.
- DBE Portal Service. – This component is responsible for the provisioning of UIs when a request is received for a service's UI. This is a service written as a DBE Service that runs inside the Servent.

To retrieve the remote UI, the method, `IDBEPortal::getUI()`, contained in the DBE Portal Client is used to retrieve the requested UI. This method returns back the URL of where the UI can be retrieved.

The remote execution of the service through the UI is UI-technology specific. As a *defacto* UI technology the computing domain partners have chosen Openlaszlo [6] as the UI technology on the recommendation of [2]. There are two main phases related to remote execution of a service when Openlaszlo is considered.

- Execution between the UI and servent: this accomplished using Openlaszlo-specific JavaRPC [7] that provides an execution binding between the Openlaszlo engine and the servent.
- Asynchronous execution between the servent and service: this is performed using standard servent and APA API's.

[3] The invocation is asynchronous between client UI and servent. However execution is not fully asynchronous due to the current P2P system implementation between servents and services.

These two tasks of remote UI retrieval and remote service execution are detailed further in the following sections (5.1, 5.2).

5.1 Remote UI Retrieval

For the retrieval of a user interface to occur the following steps (illustrated in **Figure 13** and **Figure 14**) take place:

1. The user searches for a service using the Query Formulator-Semantic Discovery Tool (QF-SDT)[4] that is displayed in the user's web browser. The service could be hosted on a remote servent.
2. The QF-SDT served by portal and displayed by the user's web browser takes the request and searches the Semantic Registry (SR) for matching services using the Portal Toolkit (PT)
3. A list of matching service offerings (Service Manifests (SM)) are displayed via the QF-SDT interface
4. The user selects a service interesting him/her.
5. The PT downloads the service proxy on the local servent corresponding to the user selected service.
6. The PT retrieves the service's UI using the service's proxy.
7. This UI is mirrored at the local portal using the PT.
8. The QF-SDT is returned a URL to the mirrored UI on the local portal and displays that UI in the user's web browser using the QF-SDT.

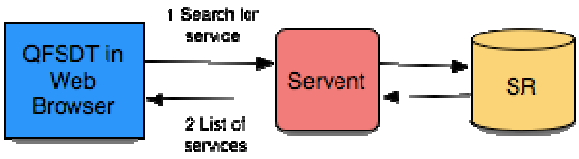


Figure 13 Remote UI Retrieval

[4] The Query Formulator/ Semantic Discovery Tool mentioned here is an Openlaszlo port of the DBE Studio QF-SDT eclipse plugin. Both have been implemented by TUC.

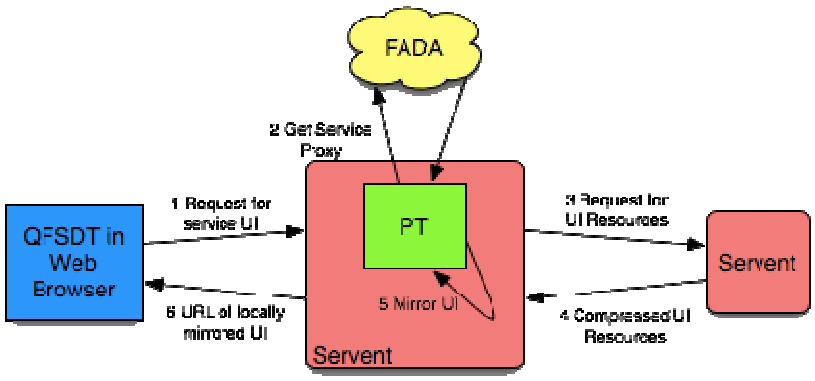


Figure 14 Remote UI Retrieval

5.2 Remote Service Execution

From what was shown in 4.2, it can be shown that either the user authenticates itself to the DIS and then invokes directly a particular service or that the DBE Portal authenticates on behalf of the user within a SME domain and relays service invocations to and from the user. The first case is typically where a user of the DBE does not need or can not run the DBE applications. In this case another means of interacting with the DBE needs to be provided. Figure 15 is an overview of how this process of a single user invoking a service may operate:

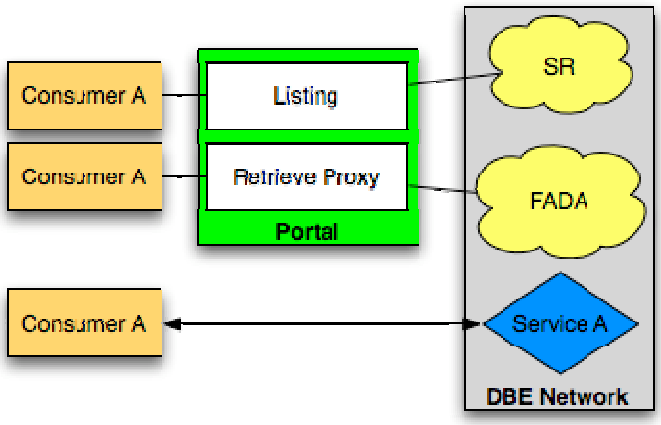


Figure 15 DBE Portal Single User Service Invocation

The task of executing a service becomes more complicated when authentication is performed on behalf of a user by the DBE Portal. In this case it is necessary for the DBE Portal to maintain a state table of all invocations made from the authenticated users. In this case the DBE Portal then becomes a service relay. This functionality can then be likened to that of a NAT. Within the Portal

Toolkit service execution through a UI is performed in the following fashion and illustrated in Figure 16:

1. The user has a UI to interact with. This UI was acquired from the remote UI retrieval process (5.1).
2. The user presses a button on the UI which calls the PT with:
    a. SMID
    b. Target method name
    c. Method parameters
3. This invocation is passed on to the Openlaszlo engine on the local servent via JavaRPC to a PT component within the Openlaszlo engine.
4. To create the remote call to the service by the PT, the PT needs to discover what are the parameter types – this is done by retrieving and reading the SDL types contained in the SM.
5. The PT invokes the Client Side Servent (CSS)
6. The CSS then invokes the remote service using servent infrastructure and the corresponding values returned to the Openlaszlo client UI via the same JavaRPC mechanism.

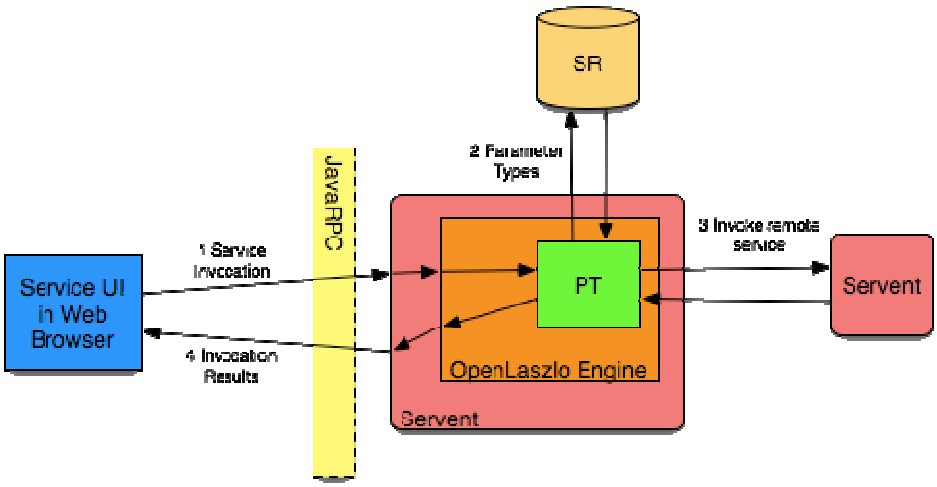


Figure 16 Remote Service Execution

5.3 Portal Toolkit Logical Structure

The portal toolkit is split into two logical parts. The first component is one which sits inside the openlaszlo engine, the DBE Portal Client. This component is shown below in the following Unified Modelling Language (UML) class diagram:

Figure 17 DBE Portal Client Class Diagram

The second logical part of the DBE Portal Toolkit is the DBE Portal service that runs as a service on the Servent. Its UML class diagram is shown as follows:

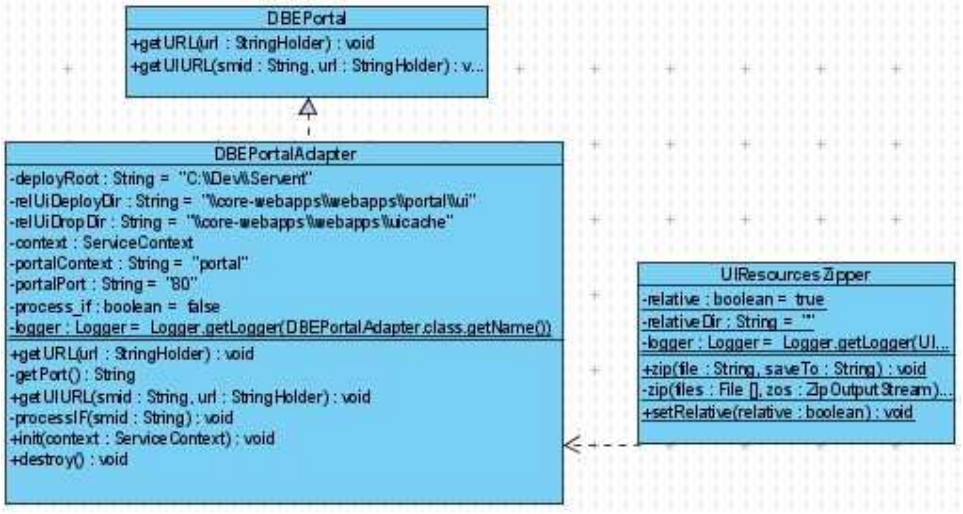


Figure 18 DBE Portal Service Class Diagram

5.4 Portal Toolkit Pending Issues

During the implementation and testing of the DBE Portal Toolkit an issue of poor latency and performance was experienced. The symptom noted initially was service UI's timing out when RPC calls were executed to the backend service associated with the UI. On further investigation, it was found that a synchronous call to the P2P system (Federated Advanced Directory Architecture, FADA) API was causing a bottleneck and was impacted by the propagation delay when P2P system nodes propagate search requests for services. As a temporary workaround the DBE Portal and its toolkit can only use one P2P system node in order to reduce latency. This issue has been raised [9] and will be resolved in the next version of FADA scheduled for May 2006.

6 DBE Portal Interdependencies

There are a number of components external to the DBE Portal on which it is dependent on. Synchronisation is required between the DBE Portal and these to ensure an approach that is well integrated and utilises each component to its maximum potential and hopefully beyond. The components on which the DBE Portal is dependent (illustrated in Figure 19) on are as follows:

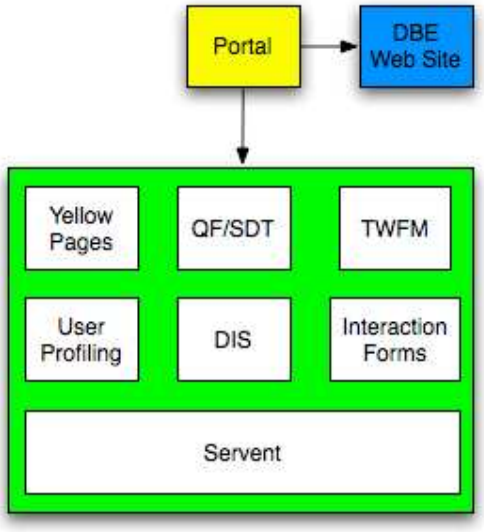


Figure 19 DBE Portal Interdependencies

- DIS – In order for the DBE Portal to provide identity registration, validation and management, the Portal will require interfaces to the DIS. The partner responsible for this component is TCD.
- Servent – This component will be required in order to provide to the DBE Portal information regarding its IP address visibility. The DBE Portal may also need the activation of the servent's underlying servlet engine that provides for servlets and Java Server Pages (JSP). The partner responsible for this component is SUN.
- User profiling – If supported by this component, DBE user preferences pertaining to the DBE Portal can be saved to this service. The partner responsible for this component is FZI.
- The "DBE web site" – It would be desirable to link all portals to the DBE web site where more general purpose documentation is available on the DBE. Also hosted at this web site are user forums where users of the DBE can interact with each other. The partner responsible for this component is Intel.

- "Yellow pages" – To be investigated. The partner responsible for this component is Soluta.
- Interaction forms – Interaction forms will provide template UIs that a SME can customise if the SME does have the required skills to author a UI on their own behalf. The partner responsible for this component is Soluta.
- Query Formulator/Semantic Discovery Tool – This component will provide the means to search and browse the contents of the DBE network for services. An Openlaszlo version of it has been integrated into the portal. The partner responsible for this component is TUC.
- Transactional Workflow Manager – This component's interfaces will be required if long running transactions and notification of their completion is to be supported in the DBE Portal. The partner responsible for this component is University of Surrey.

7 References

[1] DBE Desktop:
http://sourceforge.net/project/showfiles.php?group_id=143906&package_id=167564

[2] SUN Microsystems; M20.1: “Draft User Interface Specification”. Available from http://www.digital-ecosystem.org/.

[3] FZI; Del 7.2: “Initial Description of Profiling mechanism design and rationale with respect to one or two use cases”. Available from http://www.digital-ecosystem.org/.

[4] Folksonomy; Vanderwal, T. (2005). "Off the Top: Folksonomy Entries";
http://www.vanderwal.net/random/category.php?cat=153

[5] STUN; “Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”; http://www.ietf.org/rfc/rfc3489.txt

[6] Openlaszlo; http://www.openlaszlo.org

[7] JavaRPC; http://www.laszlosystems.com/lps-3.1.1/docs/guide/rpc-javarpc.html

[8] DBE Servent; http://swallow.sf.net

[9] FADA latency issue;
http://sourceforge.net/tracker/index.php?func=detail&aid=1415453&group_id=138511&atid=740885

[10] Dynamic Host Configuration Protocol (DHCP); http://www.ietf.org/rfc/rfc2131.txt

[11] Internet Protocol (IP); http://www.ietf.org/rfc/rfc791.txt

[12] The IP Network Address Translator (NAT); http://www.ietf.org/rfc/rfc1631.txt

8 Glossary

Acronym	Definition
APA	Abstract Protocol Adapter.
API	Application Programming Interface.
BML	Business Modelling Language.
CMS	Content Management System.
CSS	Client Side Servent.
DBE	Digital Business Ecosystem.
DIS	Distributed Identity Service.
DSS	Distributed Storage Service.
EvE	Evolutionary Environment.
ExE	Execution Environment.
FADA	Federated Advanced Directory Architecture.
IP	Internet Protocol.
ISP	Internet Service Provider.
JDK	Java Development Kit.
JSP	Java Server Pages.
KB	Knowledge Base.
NAT	Network Address Translator.
P2P	Peer-To-Peer.
PT	Portal Toolkit.
QF-SDT	Query Formulator/ Service Discovery Tool.
SDL	Service Definition Language.
SM	Service Manifest.
SME	Small-to-Medium Enterprise.
SR	Semantic Registry.
SSS	Server Side Servent.
TWFM	Transactional Workflow Manager.
UDDI	Universal Description, Discovery and Integration.
UI	User Interface.
UML	Unified Modelling Language.
WWW	World Wide Web.

9 Appendix

JavaDoc for the Portal Toolkit

Note: These interfaces and classes listed below are subject to future change and refactoring where necessary

9.1 org.dbe.toolkit.portal
Class DBEPortal

```
java.lang.Object
  └org.dbe.toolkit.portal.DBEPortal
```

All Implemented Interfaces:

IDBEPortal

---

public class **DBEPortal**
extends java.lang.Object
implements IDBEPortal

Author:

andy-edmonds

Field Summary

private org.sun.dbe.ClientHelper	**helper**

Constructor Summary

DBEPortal()

Method Summary

java.lang.Object[]	**execute**(java.lang.String smid, java.lang.String methodName, java.lang.Object[] parameters) Execute the RPC call from the UI
java.lang.Object	**executeIF**(java.lang.String smid, java.util.HashMap params) Get the IF UI of the corresponding SMID, supply the IF information and return the URL of the IF UI
java.lang.String	**getUI**(java.lang.String smid) Get the UI of the corresponding SMID and return the URL of it

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.1.1 helper

```
private org.sun.dbe.ClientHelper helper
```

Constructor Detail

9.1.2 DBEPortal

```
public DBEPortal()
```

Method Detail

9.1.3 execute

```
public java.lang.Object[] execute(java.lang.String smid,
                                  java.lang.String methodName,
                                  java.lang.Object[] parameters)
```

Description copied from interface: IDBEPortal
Execute the RPC call from the UI
Specified by:
execute in interface IDBEPortal
Parameters:
smid - the SMID of the service to execute
methodName - the name of the method to invoke
parameters - the list of parameters to supply to the method invocation
Returns:
returns a list containing the results; 1st element is the return value of ServiceProxy.invoke
See Also:
IDBEPortal

9.1.4 getUI

```
public java.lang.String getUI(java.lang.String smid)
```

Description copied from interface: IDBEPortal
Get the UI of the corresponding SMID and return the URL of it
Specified by:
getUI in interface IDBEPortal
Parameters:
smid - the SMID of the service that will supply the service UI
Returns:
a URL represented as a String pointing to the UI that the client can use
See Also:
IDBEPortal

9.1.5 executeIF

```
public java.lang.Object executeIF(java.lang.String smid,
                                  java.util.HashMap params)
```

Description copied from interface: IDBEPortal
Get the IF UI of the corresponding SMID, supply the IF information and return the URL of the IF UI
Specified by:

executeIF in interface IDBEPortal
Parameters:
smid - the SMID of the service that will supply the service UI
Returns:
a URL represented as a String pointing to the UI that the client can use
See Also:
IDBEPortal

9.2 org.dbe.toolkit.portal.service
Interface DBEPortal

All Known Implementing Classes:
DBEPortalAdapter

---

public interface **DBEPortal**
Author:
andy-edmonds

Method Summary

void	**getUIURL**(java.lang.String smid, javax.xml.rpc.holders.StringHolder url)
void	**getURL**(javax.xml.rpc.holders.StringHolder url)

Method Detail

9.2.1 getURL

```
public void getURL(javax.xml.rpc.holders.StringHolder url)
            throws java.rmi.RemoteException
```

Parameters:
url - the return value - this is the URL where the portal is at
Throws:
java.rmi.RemoteException

9.2.2 getUIURL

```
public void getUIURL(java.lang.String smid,
                     javax.xml.rpc.holders.StringHolder url)
              throws java.rmi.RemoteException
```

Parameters:
smid - the SMID of the requested UI
url - the return value - this is the URL where the UI is at
Throws:
java.rmi.RemoteException

9.3 org.dbe.toolkit.portal.service
Class DBEPortalAdapter

```
java.lang.Object
  └org.dbe.toolkit.portal.service.DBEPortalAdapter
```

All Implemented Interfaces:

org.dbe.servent.Adapter, DBEPortal

---

public class **DBEPortalAdapter**
extends java.lang.Object
implements DBEPortal, org.dbe.servent.Adapter
Author:
andy-edmonds

Field Summary

private org.dbe.servent.ServiceContext	**context**
private java.lang.String	**deployRoot**
private static org.apache.log4j.Logger	**logger**
private java.lang.String	**portalContext**
private java.lang.String	**portalPort**
private boolean	**process_if**
private java.lang.String	**relUiDeployDir**
private java.lang.String	**relUiDropDir**

Constructor Summary

DBEPortalAdapter()

Method Summary

void	**destroy**()
private java.lang.String	**getPort**() Gets the port on which the portal runs on
void	**getUIURL**(java.lang.String smid, javax.xml.rpc.holders.StringHolder url) Zips up the requested UI (corresponding to the SMID) and returns back the URL of where it is available
void	**getURL**(javax.xml.rpc.holders.StringHolder url) Gets the URL where the portal runs at
void	**init**(org.dbe.servent.ServiceContext context)
private void	**processIF**(java.lang.String smid)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.3.1 deployRoot

```
private java.lang.String deployRoot
```

9.3.2 relUiDeployDir

```
private java.lang.String relUiDeployDir
```

9.3.3 relUiDropDir

```
private java.lang.String relUiDropDir
```

9.3.4 context

```
private org.dbe.servent.ServiceContext context
```

9.3.5 portalContext

```
private java.lang.String portalContext
```

9.3.6 portalPort

```
private java.lang.String portalPort
```

9.3.7 process_if

```
private boolean process_if
```

9.3.8 logger

```
private static org.apache.log4j.Logger logger
```

Constructor Detail

9.3.9 DBEPortalAdapter

```
public DBEPortalAdapter()
```

Method Detail

9.3.10 getURL

```
public void getURL(javax.xml.rpc.holders.StringHolder url)
            throws java.rmi.RemoteException
```

Gets the URL where the portal runs at

Specified by:

`getURL` in interface `DBEPortal`

Parameters:

`url` - the return value - the portal's URL

Throws:

`java.rmi.RemoteException`

9.3.11 getPort

```
private java.lang.String getPort()
```

Gets the port on which the portal runs on

Returns:

the port number

9.3.12 getUIURL

```
public void getUIURL(java.lang.String smid,
                     javax.xml.rpc.holders.StringHolder url)
              throws java.rmi.RemoteException
```

Zips up the requested UI (corresponding to the SMID) and returns back the URL of where it is available

Specified by:

`getUIURL` in interface `DBEPortal`

Parameters:

`smid` - the SMID of the service which implements the requested UI

`url` - a return value - the URL of the zip file containing the UI

Throws:

`java.rmi.RemoteException`

9.3.13 processIF

```
private void processIF(java.lang.String smid)
```

9.3.14 init

```
public void init(org.dbe.servent.ServiceContext context)
```

Specified by:

`init` in interface `org.dbe.servent.Adapter`

9.3.15 destroy

```
public void destroy()
```

Specified by:

`destroy` in interface `org.dbe.servent.Adapter`

9.4 org.dbe.toolkit.portal.client
Class DBEPortalClient

```
java.lang.Object
  └org.dbe.toolkit.portal.client.DBEPortalClient
```

---

public class **DBEPortalClient**
extends java.lang.Object
Author:
andy-edmonds

Field Summary

private static org.apache.log4j.Logger	**logger**
(package private) java.lang.String	**smid**

Constructor Summary

DBEPortalClient()
DBEPortalClient(java.lang.String smid)

Method Summary

java.lang.Object[]	**executeIF**(java.lang.String smId, java.util.Vector paramNames, java.util.Vector paramValues) Invokes the interaction form service
java.lang.String	**getUI**(java.lang.String smid) Returns the URL of where the requested UI or IF UI is located on the Local servent
java.util.List	**invoke**(java.lang.String smid, java.lang.String methodName, java.util.Vector parameters) Invokes a DBE service

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.4.1 smid

java.lang.String **smid**

9.4.2 logger

private static org.apache.log4j.Logger **logger**

Constructor Detail

9.4.3 DBEPortalClient

public **DBEPortalClient**()

9.4.4 DBEPortalClient

```
public DBEPortalClient(java.lang.String smid)
```

Parameters:

`smid` - The SMID the client is to interact with

Method Detail

9.4.5 invoke

```
public java.util.List invoke(java.lang.String smid,
                             java.lang.String methodName,
                             java.util.Vector parameters)
```

Invokes a DBE service

Parameters:

`smid` - SMID of the target service to invoke

`methodName` - method of the servuce to invoke

`parameters` - a list of parameter values, ordered as the method signature expects them

Returns:

an ordered list of return results, the first entry in this list is always the return value of Workspace.invoke()

9.4.6 getUI

```
public java.lang.String getUI(java.lang.String smid)
```

Returns the URL of where the requested UI or IF UI is located on the Local servent

Parameters:

`smid` - The SMID of the service which has the requested UI or IF UI

Returns:

The URL of the UI or IF UI

9.4.7 executeIF

```
public java.lang.Object[] executeIF(java.lang.String smId,
                                    java.util.Vector paramNames,
                                    java.util.Vector paramValues)
```

Invokes the interaction form service

Parameters:

`paramNames` - a list of the names of the types of the parameters

`paramValues` - a list of the values of the types of the parameters

Returns:

an ordered list of return results, the first entry in this list is always the return value of Workspace.invoke()

9.5 org.dbe.toolkit.portal.service.ui
Class DBEPortalUI

```
java.lang.Object
  └org.dbe.toolkit.proxyframework.ui.AbstractServiceUI
      └org.dbe.toolkit.portal.service.ui.DBEPortalUI
```

All Implemented Interfaces:

org.dbe.toolkit.proxyframework.ui.ServiceUI

public class **DBEPortalUI**
extends org.dbe.toolkit.proxyframework.ui.AbstractServiceUI
Author:
andy-edmonds

Field Summary

(package private) org.dbe.toolkit.proxyframework.Workspace	ws

Constructor Summary

DBEPortalUI(org.dbe.toolkit.proxyframework.Workspace arg0)

Method Summary

java.lang.String	startUI()

Methods inherited from class org.dbe.toolkit.proxyframework.ui.AbstractServiceUI
getWorkspace

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.5.1 ws

org.dbe.toolkit.proxyframework.Workspace **ws**

Constructor Detail

9.5.2 DBEPortalUI

public **DBEPortalUI**(org.dbe.toolkit.proxyframework.Workspace arg0)
Parameters:
arg0 -

Method Detail

9.5.3 startUI

public java.lang.String **startUI**()
See Also:
ServiceUI.startUI()

9.6 org.dbe.toolkit.portal.service.ui
Class DBEPortalUIFactory

```
java.lang.Object
  └org.dbe.toolkit.portal.service.ui.DBEPortalUIFactory
```

All Implemented Interfaces:

org.dbe.toolkit.proxyframework.ui.UIFactory

Deprecated. *This class implements a deprecated method of UI retrieval*

public class **DBEPortalUIFactory**
extends java.lang.Object
implements org.dbe.toolkit.proxyframework.ui.UIFactory

Author:

andy-edmonds

Field Summary

(package private) static java.lang.String	**JFRAME** **Deprecated.**
(package private) static java.lang.String	**WEB** **Deprecated.**

Constructor Summary

DBEPortalUIFactory() **Deprecated.**

Method Summary

org.dbe.toolkit.proxyframework.ui.ServiceUI	**createServiceUI**(java.lang.String uiType, org.dbe.toolkit.proxyframework.Workspace ws) **Deprecated.**
java.lang.String[]	**getUITypes**() **Deprecated.**

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.6.1 JFRAME

```
static final java.lang.String JFRAME
```

Deprecated.

See Also:

Constant Field Values

9.6.2 WEB

`static final java.lang.String` **`WEB`**

Deprecated.

See Also:

Constant Field Values

Constructor Detail

9.6.3 DBEPortalUIFactory

`public` **`DBEPortalUIFactory`**`()`

Deprecated.

Method Detail

9.6.4 createServiceUI

```
public org.dbe.toolkit.proxyframework.ui.ServiceUI
createServiceUI(java.lang.String uiType,

org.dbe.toolkit.proxyframework.Workspace ws)
                                                    throws
org.dbe.toolkit.proxyframework.ui.UnsupportedUIException
```

Deprecated.

Specified by:

`createServiceUI` in interface `org.dbe.toolkit.proxyframework.ui.UIFactory`

Throws:

`org.dbe.toolkit.proxyframework.ui.UnsupportedUIException`

See Also:

`UIFactory.createServiceUI(java.lang.String, org.dbe.toolkit.proxyframework.Workspace)`

9.6.5 getUITypes

`public java.lang.String[]` **`getUITypes`**`()`

Deprecated.

Specified by:

`getUITypes` in interface `org.dbe.toolkit.proxyframework.ui.UIFactory`

See Also:

`UIFactory.getUITypes()`

9.7 org.dbe.toolkit.portal.execution

Class ExecutionDelegate

```
java.lang.Object
  └org.dbe.toolkit.portal.execution.ExecutionDelegate
```

---

public class **ExecutionDelegate**
extends java.lang.Object

Author:

andy-edmonds

Field Summary

private org.sun.dbe.ClientHelper	**helper**

private java.util.Vector	**holderObjects**
private static org.apache.log4j.Logger	**logger**
private java.util.Vector	**outParamHolders**
private java.lang.Class[]	**outParamHoldersArray**
private java.lang.String	**smid**
private org.dbe.toolkit.proxyframework.Workspace	**ws**

Constructor Summary

ExecutionDelegate(org.sun.dbe.ClientHelper helper, java.lang.String smid)

Method Summary

private java.lang.String	**dumpSDL**(java.lang.String sdlStr) Takes a SDL file and dumps it to string
java.lang.Object[]	**execute**(java.lang.String methodName, java.lang.Object[] parameters) Executes the method with supplied parameters
java.lang.Object	**executeIF**(java.util.HashMap parameters)
private java.lang.Object	**getHolderValue**(java.lang.Object holder) Gets the value of a holder object based on type intros
private java.lang.Class	**getInClass**(org.dbe.studio.editor.sdl.Type type) Finds the java @see Class object of the type supplied
private java.lang.Object[]	**getInvocationResults**(java.lang.Object ret, java.lang.Object[] parameters) Merges out paramters and result of Workspace.invok one array
private java.lang.Class[]	**getMethodParametersTypes**(java.lang.String method Gets the parameter types for a particular method
private java.lang.Object[]	**getMethodParametersValues**(java.lang.Object[] par
private java.lang.Class	**getOutClass**(org.dbe.studio.editor.sdl.Type type, Gets the @see Class object of the supplied type
private org.dbe.toolkit.proxyframework.Workspace	**getWorkspace**() Gets a Workspace object

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.7.1 helper

```
private org.sun.dbe.ClientHelper helper
```

9.7.2 ws

```
private org.dbe.toolkit.proxyframework.Workspace ws
```

9.7.3 outParamHolders

```
private java.util.Vector outParamHolders
```

9.7.4 outParamHoldersArray

```
private java.lang.Class[] outParamHoldersArray
```

9.7.5 holderObjects

```
private java.util.Vector holderObjects
```

9.7.6 smid

```
private java.lang.String smid
```

9.7.7 logger

```
private static org.apache.log4j.Logger logger
```

Constructor Detail

9.7.8 ExecutionDelegate

```
public ExecutionDelegate(org.sun.dbe.ClientHelper helper,
                         java.lang.String smid)
```

Parameters:

helper - The ClientHelper object used to get proxies from the servent
smid - The SMID to execute

Method Detail

9.7.9 executeIF

```
public java.lang.Object executeIF(java.util.HashMap parameters)
```

Parameters:

parameters -

Returns:

9.7.10 execute

```
public java.lang.Object[] execute(java.lang.String methodName,
                                  java.lang.Object[] parameters)
```

Executes the method with supplied parameters

Parameters:

`methodName` - Name of the method to execute

`parameters` - parameters to supply to the the method

Returns:

Results of the exection

9.7.11 getWorkspace

```
private org.dbe.toolkit.proxyframework.Workspace getWorkspace()
```

Gets a Workspace object

Returns:

The workspace object

9.7.12 getMethodParametersTypes

```
private java.lang.Class[] getMethodParametersTypes(java.lang.String methodName)
                                                   throws java.lang.Exception
```

Gets the parameter types for a particular method

Parameters:

`methodName` - The method to get the parameters types

Returns:

A list of method parameter types in the order of the method's signature

Throws:

`java.lang.Exception`

9.7.13 dumpSDL

```
private java.lang.String dumpSDL(java.lang.String sdlStr)
```

Takes a SDL file and dumps it to string

Parameters:

`sdlStr` - the absolute path of a SDL file

Returns:

The SDL as a string

9.7.14 getInClass

```
private java.lang.Class getInClass(org.dbe.studio.editor.sdl.Type type)
                           throws java.lang.Exception
```

Finds the java @see Class object of the type supplied

Parameters:

`type` - type to get Class object of

Returns:

Class object supplied

Throws:

`java.lang.Exception`

9.7.15 getOutClass

```
private java.lang.Class getOutClass(org.dbe.studio.editor.sdl.Type type,
                                    int j)
                             throws java.lang.Exception
```

Gets the @see Class object of the supplied type

Parameters:

`type` - Type to find the @see Class object of

`j` - a reference counter

Returns:

Class object of the supplied type

Throws:

`java.lang.Exception`

9.7.16 getMethodParametersValues

```
private java.lang.Object[]
getMethodParametersValues(java.lang.Object[] parameters)
```

Parameters:

`parameters` -

Returns:

9.7.17 getInvocationResults

```
private java.lang.Object[] getInvocationResults(java.lang.Object ret,
                                                java.lang.Object[] parameters)
```

Merges out paramters and result of Workspace.invoke() into one array

Parameters:

`ret` - Return value of Workspaace.invoke()

`parameters` - out parameters of the rpc call

Returns:

merged array

9.7.18 getHolderValue

```
private java.lang.Object getHolderValue(java.lang.Object holder)
                                 throws java.lang.Exception
```

Gets the value of a holder object based on type introspection

Parameters:

`holder` - Holder to get the value of

Returns:

the value of the holder

Throws:

`java.lang.Exception`

9.8 org.dbe.toolkit.portal
Interface IDBEPortal

All Known Implementing Classes:

DBEPortal

---

public interface **IDBEPortal**

Author:

andy-edmonds

Method Summary

java.lang.Object[]	**execute**(java.lang.String smid, java.lang.String methodName, java.lang.Object[] parameters) Execute the RPC call from the UI
java.lang.Object	**executeIF**(java.lang.String smId, java.util.HashMap params) Get the IF UI of the corresponding SMID, supply the IF information and return the URL of the IF UI
java.lang.String	**getUI**(java.lang.String smid) Get the UI of the corresponding SMID and return the URL of it

Method Detail

9.8.1 execute

```
public java.lang.Object[] execute(java.lang.String smid,
                                  java.lang.String methodName,
                                  java.lang.Object[] parameters)
```

Execute the RPC call from the UI

Parameters:

`smid` - the SMID of the service to execute

`methodName` - the name of the method to invoke

`parameters` - the list of parameters to supply to the method invocation

Returns:

returns a list containing the results; 1st element is the return value of ServiceProxy.invoke

9.8.2 getUI

```
public java.lang.String getUI(java.lang.String smid)
```

Get the UI of the corresponding SMID and return the URL of it

Parameters:

`smid` - the SMID of the service that will supply the service UI

Returns:

a URL represented as a String pointing to the UI that the client can use

9.8.3 executeIF

```
public java.lang.Object executeIF(java.lang.String smId,
                                  java.util.HashMap params)
```

Get the IF UI of the corresponding SMID, supply the IF information and return the URL of the IF UI

Returns:

a URL represented as a String pointing to the customised IF UI that the client can use

9.9 org.dbe.toolkit.portal.network
Class SDLRetriever

```
java.lang.Object
  └org.dbe.toolkit.portal.network.SDLRetriever
```

public class **SDLRetriever**

extends java.lang.Object
Author:
andy-edmonds

Field Summary

private static org.apache.log4j.Logger	**logger**

Constructor Summary

SDLRetriever()

Method Summary

java.lang.String	**getSDL**(java.lang.String smid) Gets the SDL of the supplied SMID

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.9.1 logger

```
private static org.apache.log4j.Logger logger
```

Constructor Detail

9.9.2 SDLRetriever

```
public SDLRetriever()
```

Method Detail

9.9.3 getSDL

```
public java.lang.String getSDL(java.lang.String smid)
```

Gets the SDL of the supplied SMID

Parameters:

`smid` - SMID to get the SDL of

Returns:

SDL as a string value

9.10 org.dbe.toolkit.portal.ui.tools
Class UIResourcesUnzipper

```
java.lang.Object
  └org.dbe.toolkit.portal.ui.tools.UIResourcesUnzipper
```

---

public class **UIResourcesUnzipper**
extends java.lang.Object
Author:
andy-edmonds

Field Summary

private static org.apache.log4j.Logger	**logger**

Constructor Summary

UIResourcesUnzipper()

Method Summary

static void	**unzip**(java.lang.String zipName, java.lang.String extractionPath) Extracts a zip file

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.10.1 logger

```
private static org.apache.log4j.Logger logger
```

Constructor Detail

9.10.2 UIResourcesUnzipper

```
public UIResourcesUnzipper()
```

Method Detail

9.10.3 unzip

```
public static void unzip(java.lang.String zipName,
                         java.lang.String extractionPath)
                  throws java.io.IOException
```

Extracts a zip file

Parameters:

`zipName` - the absolute name of the zip file

`extractionPath` - the absolute path to where the zip file is to be extracted to

Throws:

`java.io.IOException`

9.11 org.dbe.toolkit.portal.ui.tools
Class UIResourcesZipper

```
java.lang.Object
  └org.dbe.toolkit.portal.ui.tools.UIResourcesZipper
```

public class **UIResourcesZipper**
extends java.lang.Object

Author:
andy-edmonds

Field Summary

private static org.apache.log4j.Logger	**logger**
private static boolean	**relative**
private static java.lang.String	**relativeDir**

Constructor Summary

UIResourcesZipper()

Method Summary

static void	**setRelative**(boolean relative) Sets archive creation to 'relative'
private static void	**zip**(java.io.File[] files, java.util.zip.ZipOutputStream zos)
static void	**zip**(java.lang.String file, java.lang.String saveTo) create a zip file from a list of files it'll recursively zip if a file's a directory

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.11.1 relative

```
private static boolean relative
```

9.11.2 relativeDir

```
private static java.lang.String relativeDir
```

9.11.3 logger

```
private static org.apache.log4j.Logger logger
```

Constructor Detail

9.11.4 UIResourcesZipper

```
public UIResourcesZipper()
```

Method Detail

9.11.5 zip

```
public static void zip(java.lang.String file,
                       java.lang.String saveTo)
```

create a zip file from a list of files it'll recursively zip if a file's a directory

Parameters:

`file` - Name of the zip file to create

`saveTo` - location to save the created zip file to

9.11.6 zip

```
private static void zip(java.io.File[] files,
                        java.util.zip.ZipOutputStream zos)
```

9.11.7 setRelative

```
public static void setRelative(boolean relative)
```

Sets archive creation to 'relative'

Parameters:

`relative` - a boolean to set the flag

9.12 org.dbe.toolkit.portal.ui
Class UIRetrievalDelegate

```
java.lang.Object
  └org.dbe.toolkit.portal.ui.UIRetrievalDelegate
```

---

public class **UIRetrievalDelegate**
extends java.lang.Object

Author:

andy-edmonds

Field Summary

private static java.lang.String	**configFile**
private org.sun.dbe.ClientHelper	**helper**
private static java.lang.String	**http**
private static org.apache.log4j.Logger	**logger**

private java.lang.String	**portNumber**
private java.lang.String	**uiCacheDir**
private java.lang.String	**uiCacheWebContext**

Constructor Summary

UIRetrievalDelegate(org.sun.dbe.ClientHelper helper)

Method Summary

private java.lang.String	**getCacheLocalUI**(java.lang.String smid) Downloads the remote UI as a zip file, extracts it and generates a URL for it
private java.lang.String	**getLocalUIURL**(java.io.File outputFile, java.lang.String smid) Unzips the recieved remote UI zip, extracts it to a web context and returns back the URL of that locally cached UI
private java.lang.String	**getPortalID**(java.lang.String smid) Gets the SMID of the portal where the requested UI is located
private java.lang.String	**getServiceUI**(java.lang.String smid) This gets the URL where the service UI is located
java.lang.String	**getUI**(java.lang.String smid) Gets the URL of a locally cached copy of the requested UI
private java.io.File	**getUIArchive**(java.lang.String smid, java.lang.String uiZipURL) Downloads the zipped UI from the remote servent
private java.lang.String	**getUIArchiveURL**(java.lang.String smid) Gets the URL of the remote UI zip file
private java.lang.String	**getUiDeploymentRoot**() Gets the location where the servent deploys its services
private java.lang.String	**getUIEntryPoint**(java.lang.String url) This gets the main entry point for an openlaszlo UI

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.12.1 configFile

private static final java.lang.String **configFile**

See Also:

Constant Field Values

9.12.2 http

`private static final java.lang.String` **`http`**

See Also:

Constant Field Values

9.12.3 helper

`private org.sun.dbe.ClientHelper` **`helper`**

9.12.4 uiCacheDir

`private java.lang.String` **`uiCacheDir`**

9.12.5 uiCacheWebContext

`private java.lang.String` **`uiCacheWebContext`**

9.12.6 portNumber

`private java.lang.String` **`portNumber`**

9.12.7 logger

`private static org.apache.log4j.Logger` **`logger`**

Constructor Detail

9.12.8 UIRetrievalDelegate

`public` **`UIRetrievalDelegate`**`(org.sun.dbe.ClientHelper helper)`

Parameters:

`helper` - The ClientHelper object used to get proxies from the servent

Method Detail

9.12.9 getUI

`public java.lang.String` **`getUI`**`(java.lang.String smid)`

Gets the URL of a locally cached copy of the requested UI

Parameters:

`smid` - the SMID of the service which has the requested UI

Returns:

the URL of the UI

9.12.10 getCacheLocalUI

`private java.lang.String` **`getCacheLocalUI`**`(java.lang.String smid)`

Downloads the remote UI as a zip file, extracts it and generates a URL for it

Parameters:

`smid` - SMID of the service implementing the requested UI

Returns:

URL of the locally cached UI

9.12.11 getUIArchiveURL

```
private java.lang.String getUIArchiveURL(java.lang.String smid)
```

Gets the URL of the remote UI zip file

Parameters:

`smid` - SMID of the service implementing the service

Returns:

URL of the remote UI zip file

9.12.12 getPortalID

```
private java.lang.String getPortalID(java.lang.String smid)
```

Gets the SMID of the portal where the requested UI is located

Parameters:

`smid` - SMID of the service implementing the UI requested

Returns:

the ID of the hosting portal

9.12.13 getUIArchive

```
private java.io.File getUIArchive(java.lang.String smid,
                                  java.lang.String uiZipURL)
```

Downloads the zipped UI from the remote servent

Parameters:

`smid` - SMID of the service that implements the UI

`uiZipURL` - the URL of the zipped UI

Returns:

a @see File object that corresponds to the zip file containing the UI

9.12.14 getUiDeploymentRoot

```
private java.lang.String getUiDeploymentRoot()
```

Gets the location where the servent deploys its services

Returns:

the deployment directory

9.12.15 getLocalUIURL

```
private java.lang.String getLocalUIURL(java.io.File outputFile,
                                       java.lang.String smid)
```

Unzips the recieved remote UI zip, extracts it to a web context and returns back the URL of that locally cached UI

Parameters:

`outputFile` - the file to unzip

`smid` - the SMID of the service implementing the UI

Returns:

the local URL of the UI

9.12.16 getServiceUI

```
private java.lang.String getServiceUI(java.lang.String smid)
```

This gets the URL where the service UI is located

Parameters:
`smid` - SMID of the service implementing the UI
Returns:
The URL of the UI

9.12.17 getUIEntryPoint

```
private java.lang.String getUIEntryPoint(java.lang.String url)
```

This gets the main entry point for an openlaszlo UI
Parameters:
`url` - The URL to extract the main entry point from
Returns:
The main entry point