Informatics for Integrating Biology and the Bedside

i2b2 Software Architecture

Workplace Framework (WORK) Cell

Document Version: *1.6.2*

i2b2 Software Version: *1.6*

Table of Contents

Document Management 4

Abstract 5

1. Overview 6

1.1 WORK Definitions, Acronyms and Abbreviations 6

1.1.1 Work Data Object (WDO) 6

1.2 Roles 6

1.3 Security 7

1.4 Scope of the System 8

1.5 Assumptions/Constraints 8

1.6 Technical Platform 8

1.6.1 Transaction 8

1.6.2 Security 8

1.6.3 Persistence 8

1.6.4 Reliability/Availability 9

1.6.5 Performance 9

2. Use Case 10

2.1 Operations 10

3. Architecture Description 12

3.1 Components and Connector View 12

3.1.1 Client-Server Style 12

3.1.1.1 Primary Presentation 12

3.1.1.2 Element Catalog 13

3.1.1.3 Design Rationale, Constraints 14

3.2 Module View type 14

3.2.1 Decomposition Style 14

3.2.1.1 Primary Presentation 14

3.2.1.2 Element Catalog 14

3.2.1.3 Context Diagram 14

3.2.2 Uses Style 15

3.2.2.1 Primary Presentation 15

3.2.2.2 Element Catalog 15

3.2.2.3 Context Diagram 16

3.2.2.4 Sequence Diagram 16

3.3 Mappings of Styles 17

4. Data View 18

4.1 Selecting the Data Source 18

4.2 Schemas within the workplace data source 20

4.2.1 Table_Access table 20

4.2.2 Workplace table 21

5. Drag and Drop Actions 22
5.1 Current 22
6. Deployment View 23
6.1 Global Overview 23
6.2 Detailed Deployment Model 23
References 24

DOCUMENT MANAGEMENT

Revision Number	Date	Author	Description of change
1.6.1	07/22/10	Janice Donahoe	Created 1.6 version of document.
1.6.2	09/23/11	Mike Mendis	Added export child

ABSTRACT

This is a software architecture document for Workplace Framework (WORK) cell. It identifies and explains important architectural elements. This document will serve the needs of stake holders to understand system concepts and give a brief summary of the use of the WORK message format.

1. OVERVIEW

The **Workplace Framework cell (WORK)** is a core i2b2 Hive cell. This cell manages project specific XML data objects for users of a given project. The project specific XML data objects originate in other views or cells, such as Ontology or Previous Query and are stored in the WORK cell as a convenience.

The project specific XML data objects in the WORK cell are organized in hierarchical structures that represent the relationships between elements. The top levels in the hierarchy are called the "**parents**" or "**roots**", with the lower levels being their "**children**". Elements occurring on the same level are known as "**siblings**". A level in a hierarchy is sometimes referred to as a "**node**".

The WORK cell both accepts new XML data objects for storage and provides a listing of those items previously stored. It also allows the user to organize, label and annotate the stored data objects however they choose to.

1.1 WORK Definitions, Acronyms and Abbreviations

1.1.1 Work Data Object (WDO)

This object holds workplace folder definitions and information about folder contents.

1.2 Roles

When and how data is presented to a user is based on their user roles, which are specified in the PM Cell. Each user will have at least two roles per user_id and product_id combination. These two roles can be further defined as a Data Protection role and a Hive Management role.

The data protection role establishes the detail of data the user can see while the hive managment role defines their level of functionality the user has in a project.The following tables summarize the roles in a hierarchical order of least to most access.

Data Protection Track	
Role	**Access Description**
DATA_OBFSC	OBFSC = Obfuscated ▪ The user can see aggregated results that are obfuscated (example: patient count). ▪ The user is limited on the number of times they can run the same query within a specified time period. If the user exceeds the maximum number of times then

	their account will be locked and only the Admin user can unlock it.
DATA_AGG	AGG = Aggregated ▪ The user can see aggregated results like the patient count. ▪ The results are not obfuscated and the user is not limited to the number of times they can run the same query.
DATA_LDS	LDS = Limited Data Set ▪ The user can see all fields except for those that are encrypted. ▪ An example of an encrypted field is the *blob fields* in the *fact* and *dimension tables*.
DATA_DEID	DEID = De-identified Data ▪ The user can see all fields including those that are encrypted. ▪ An example of an encrypted field is the *blob fields* in the *fact* and *dimension tables*.
DATA_PROT	PROT = Protected ▪ The user can see all data, including the identified data that resides in the Identity Management Cell.

Hive Management Track	
Role	**Access Description**
USER	Can create queries and access them if he/she is the owner of the query.
MANAGER	Can create queries and can access queries created by different users within the project.

Further details regarding roles can be found in the PM_Design_Document.

1.3 Security

Users may access WORK with a user-id and password combination, which is authorized through the Project Management Cell. The implementation detail of Project Management Cell is considered out-of scope to this document.

1.4 Scope of the System

Some other participants, currently outside the scope of WORK are:

- Project Management Cell

1.5 Assumptions/Constraints

The Workplace database will not contain protected health information.

1.6 Technical Platform

The product is designed to run under the following platform:

- Java 2 Standard Edition 6.0 version 16
- Oracle Server 10g database
- Xerces2 XML parser
- JBoss Application server version 4.2.2.GA
- Spring Web Framework 2.0
- Axis2 v1.1 web service (SOAP/REST messaging)

1.6.1 Transaction

The WORK system is transactional, leveraging the transaction management model of the J2EE platform.

1.6.2 Security

The application must implement basic security behaviors:

- **Authentication:** Authenticate using at least a user name and a password
- **Authorization:** User may only access categories that they are allowed to by role
- **Confidentiality:** Sensitive data must be encrypted
- **Data integrity:** Data sent across the network cannot be modified by a tier
- **Auditing:** In the later releases we may implement logging of sensitive actions

1.6.3 Persistence

This application utilizes JDBC calls to retrieve persisted data.

1.6.4 Reliability/Availability

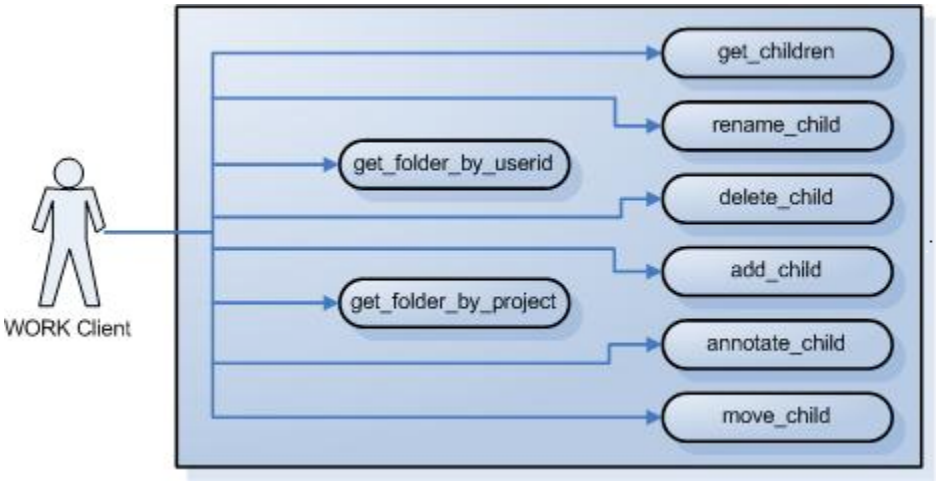
- The Reliability/Availability will be addressed through the J2EE platform
- Targeted availability is 16/7: 16 hours a day, 7 days a week
- The time left (8 hours) is reserved for any maintenance activities

1.6.5 Performance

The user authentication with project management cell must be under 1 second.

2. USE CASE

The diagram below depicts common use cases a user may perform with the WORK cell.



2.1 Operations

The WORK service is designed as a collection of operations, or use cases:

Service	Description
get_folder_by_userid	Returns a list of root folders available for a given user. These folders are displayed in a tree format. The top level of the tree consists of all the folders a particular user has permission to see as determined by his/her role.
get_folder_by_project	Returns a list of root folders available for a given project across all users in the project. These folders are displayed in a tree format. A user must have manager permission and be configured as such to see all folders in a project.
get_children	Expands any level of a folder, providing information about its contents, for a given user.
rename_child	Renames a work item (leaf or folder) in the workplace tree.
delete_child	Deletes a work item (leaf or folder) in the workplace tree.
add_child	Adds a work item (leaf or folder) in the workplace tree.
annotate_child	Annotates a work item (leaf or folder) in the workplace tree by modifying or adding a tooltip.

move_child	Moves a work item (folder) in the workplace tree and reassigns all its children.
export_child	Export a work item (leaf or folder) as XML to a local file

3. ARCHITECTURE DESCRIPTION

This section provides a description of the architecture as multiple views. Each view conveys the different attributes of the architecture.

1. Components and Connector View
    a. Client-Server Style

2. Module View
    a. Decomposition Style
    b. Uses Style

3. Data View
4. Deployment View

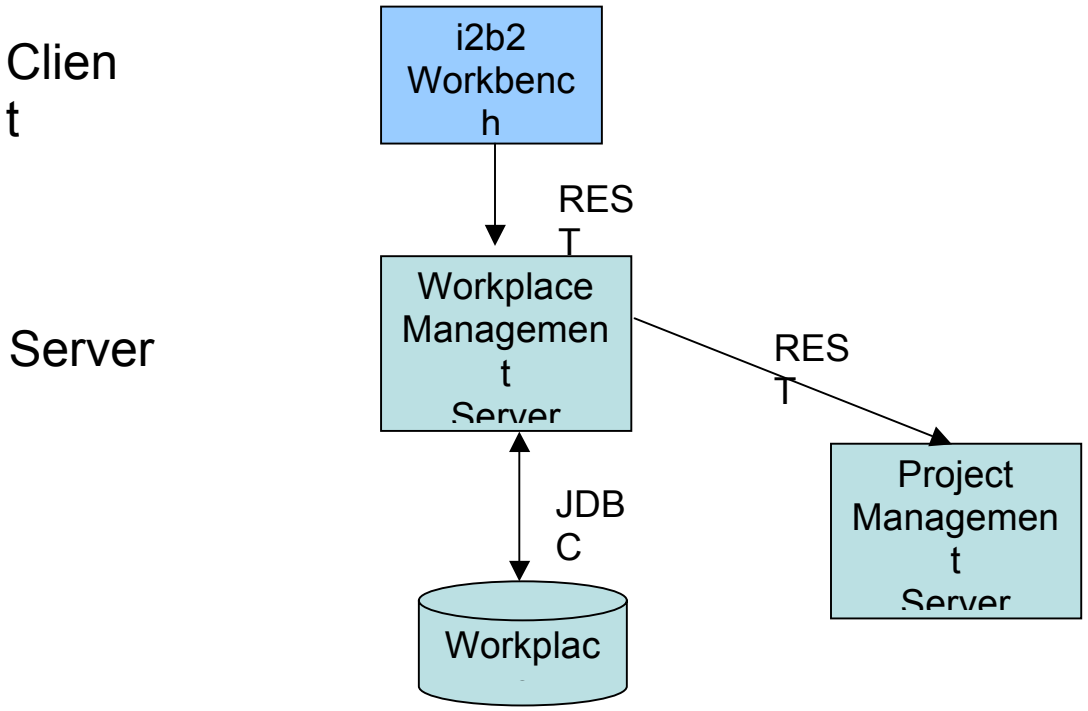
3.1 Components and Connector View

A **Component and Connector view** represents the runtime instances and the protocols of connection between the instances. The connectors represent the properties such as concurrency, protocols and information flows. The diagram shown in the *Primary Presentation* section represents the Component and Connector view for the multi-user installation. As seen in the diagram, component instances are shown in more detail with specific connectors drawn in different notations.

3.1.1 Client-Server Style

The WORK system is represented using the components and connector client-server view.

3.1.1.1 PRIMARY PRESENTATION



3.1.1.2 ELEMENT CATALOG

i2b2 Workbench	Client Component	Webservice client submits the requests to WORK Server components and renders response XML.
Workplace Framework Server	Server Component	Provides Web Service Interface for the WORK system. It supports the REST protocol. It uses Project Management server to handle user authentication.
Project Management Server	Server Component	WORK cell uses Project Management cell to authenticate user. WORK cell constructs PM request message and makes a web service call to Project Management Cell.
Workplace	Data Repository Component	This repository is a database for i2b2 workplace data.
JDBC	Query Connector	SQL query used as a connector between the WORK System and the Metadata database.
Web Service	Request	REST protocol used to communicate with the external system.

	Connector	

3.1.1.3 DESIGN RATIONALE, CONSTRAINTS

N-tier Architecture

The client-server style depicts an n-tier architecture that separates the presentation layer from business logic and data access layer.

3.2 Module View type

The module view shows how the system is decomposed into implementation units and how the functionality is allocated to these units. The layers show how modules are encapsulated and structured. The layers represent the “allowed-to-use” relation.

The following sections describe the module view using Decomposition and Uses Styles.

3.2.1 Decomposition Style

The Decomposition style presents system functionality in terms of manageable work pieces. It identifies modules and breaks them down into sub-modules and so on, until a desired level of granularity is achieved.

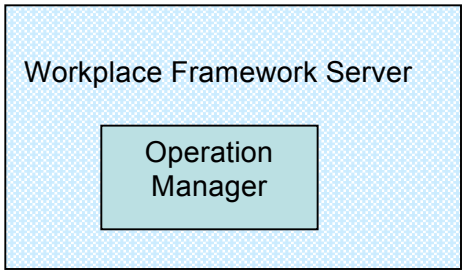
3.2.1.1 PRIMARY PRESENTATION

Workplace Framework Server	Operation Manager

3.2.1.2 ELEMENT CATALOG

Element Name	Type	Description
Operation Manager	Subsystem	This subsystem manages queries for Workplace operations.

3.2.1.3 CONTEXT DIAGRAM



3.2.2 Uses Style

The Uses style shows the relationships between modules and sub-modules. This view is very helpful for implementing, integrating and testing the system.

3.2.2.1 PRIMARY PRESENTATION

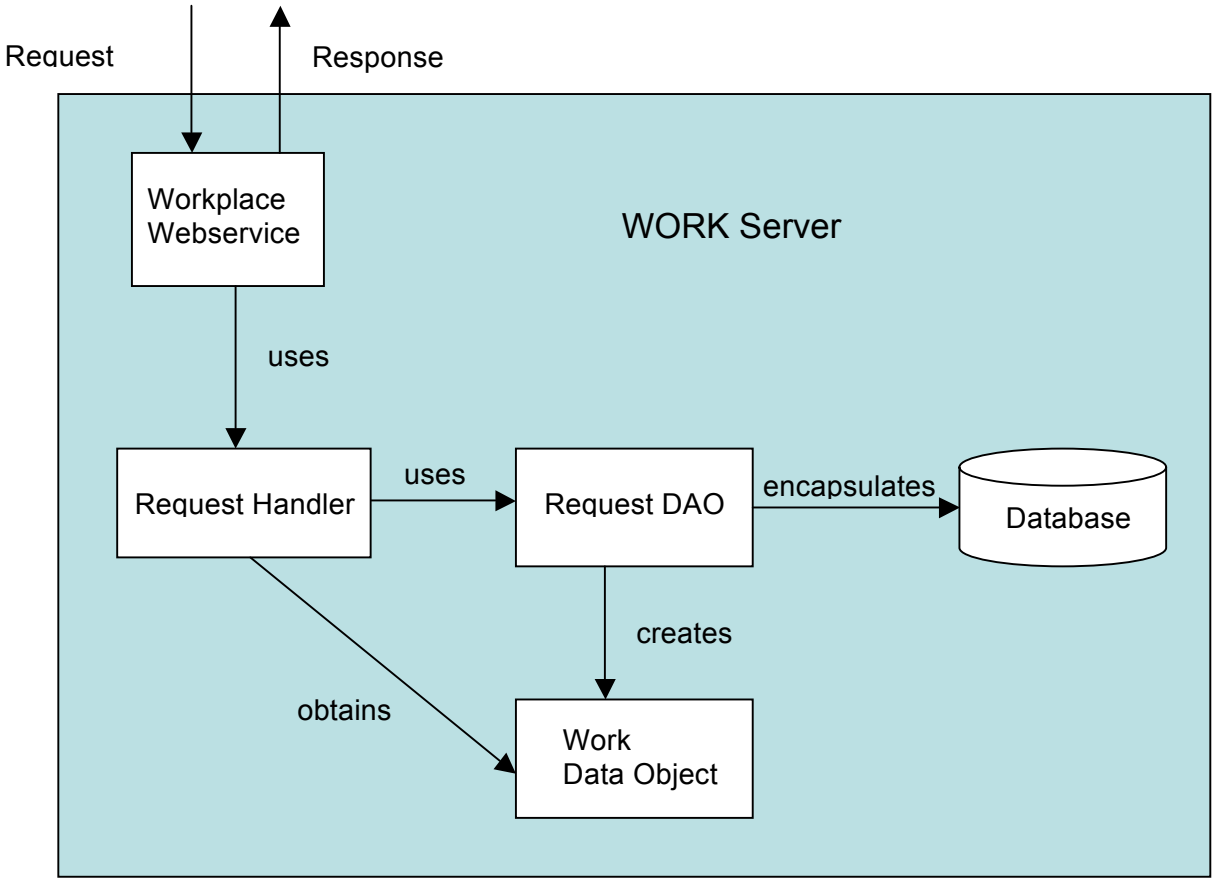
System	Segment
Workplace Framework Server	WORK Module
Operation Manager Subsystem	Workplace Webservice
	Request Handler
	Request DAO
	Work Data Object

3.2.2.2 ELEMENT CATALOG

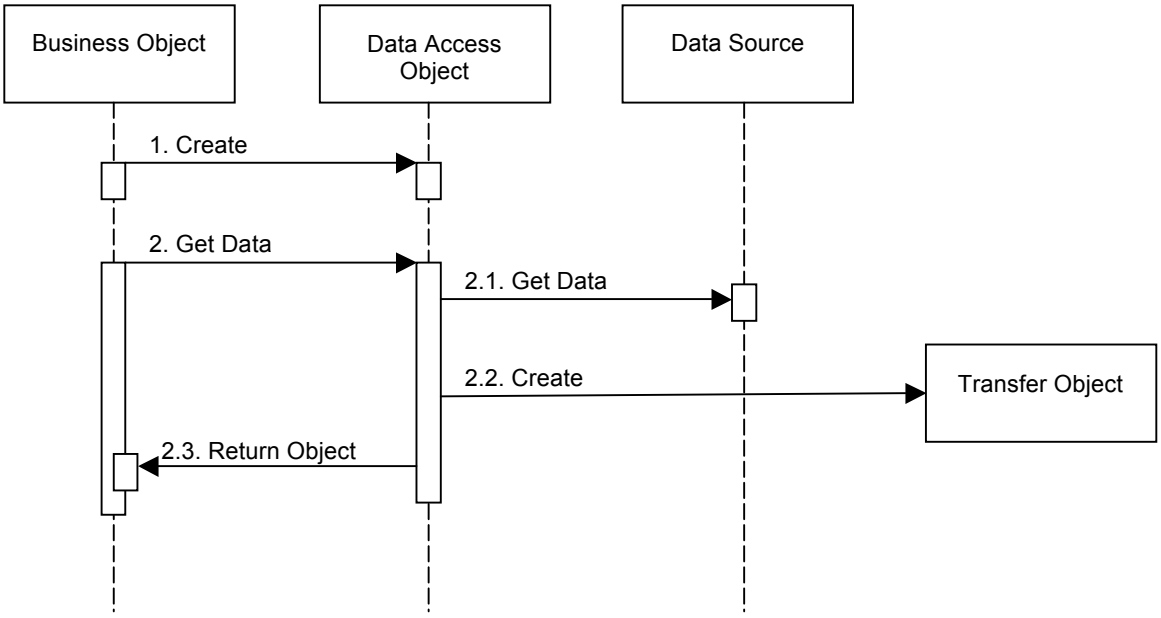
Element Name	Type	Description
WORK Module	Module	Authenticates user through PM Server System
Workplace Webservice	Communication Module	Provides web service interface to Workplace operations.
Request Handler	Business Object	Delegates Workplace requests to Data Access Object layer to perform database operations.
Request DAO	Data Access Object	Supports database query operations.

Work Data Object	Transfer Object	Object representation of persisted data

3.2.2.3 CONTEXT DIAGRAM



3.2.2.4 SEQUENCE DIAGRAM



3.3 Mappings of Styles

The following table is a mapping between the elements in the Components and Connector Client-Server view shown in section 3.1.1, and the Modules Decomposition and Uses views shown in sections 3.2.1 and 3.2.2.

The relationship shown is *is-implemented-by*, i.e. the elements from the Components and Connector view shown at the top of the table are implemented by any selected elements from the Modules views, denoted by an “X” in the corresponding cell.

	WORK Server	**Project Management Server**	**Workplace Database**
WORK Service	X	X	
Workplace Webservice	X		
Request Handler	X		
Request DAO	X		X
Work Data Object	X		

4. DATA VIEW

4.1 Selecting the Data Source

Stored Workplace data is distributed to projects through the existence of independent databases (in SQL Server) or schemas (in Oracle). These will be referred to in the rest of the text as the "persistent storage location" or PSL. These PSL's are organized so that the data from two metadata representations can be merged to a "Super" data set. While a person is working on a specific project, they will be directed to data in a PSL associated with that project.

In order to support the i2b2 project distribution strategy, the user is enrolled in numerous projects recorded within the i2b2 project management cell". The projects available to the user are returned in the web service call to the Project Management cell. The logic of selecting the correct PSL for the project is embodied in the following table:

DB_LOOKUP		
PK	**c_domain_id**	**VARCHAR(255)**
PK	**c_project_path**	**VARCHAR(255)**
PK	**c_owner_id**	**VARCHAR(255)**
	c_db_fullschema	**VARCHAR(255)**
	c_db_datasource	**VARCHAR(255)**
	c_db_servertype	**VARCHAR(255)**
	c_db_nicename	VARCHAR(255)
	c_db_tooltip	VARCHAR(255)
	c_comment	CLOB
	c_entry_date	DATE
	c_change_date	DATE
	c_status_cd	CHAR(1)

The logic for selecting the PSL is as follows:

1. There are two methods to select the correct PSL, an implicit one, and an explicit one. Both rely only on information available within the i2b2 header.

    c. The implicit one relies upon the data within the <domain> tag, the <username> tag, and the <project_id> tag.

d. The explicit one relies upon the data only within the <project_id> tag. It has the format represented as the following string:

|"DOMAIN" | "Project"\"subproject"\"sub-sub Project"\| "USER_ID"|

- ***These may not actually match the domain and username that is actually being used (since it is being built by the client), and must be checked when the PM cell is accessed.***

2. The table is meant to provide a series of default locations if ones are not specifically listed. If a project is listed in the *c_project_path* column, then that PSL may be used, otherwise, a domain source will be used.
3. If a username is listed in the *c_owner_id* column, then if the project also matches the *project_id*, the PSL in that row may be used, otherwise, a project PSL will be used, and if a project PSL does not exist, the domain PSL will be used. For example, only if the *domain|project|user_id* is an EXACT match to the entries in the database will that PSL be used.
4. The project id may have associated sub projects that will be represented as *project\sub-project\sub-sub-project* string. If a sub-project is identified, but only the project exists in the table, the project PSL would be used.
5. The project may not have an entry in the table, and in that case any project (and sub-projects) would be designated the PSL of the domain.
6. If a general domain PSL is not available in the table, and only a specific project is associated with the domain in the table, then any incoming messages not associated with that project will return an error.
7. In the table, the "@" character is used to represent the absence of an entry (rather than a blank or a null).
8. In the explicit string and in the <project_id>, and "@" can be used to optionally represent a blank.

Other columns are specified as follows:

9. The column *c_db_fullschema* is used to contain the path to a table when the data source is used. Software is written so that the absence of the delimiter (usually a ".") does not need to be explicitly stated.
10. The column *c_db_datasource* is used to contain a short string that represents a data source configured in some other location.
11. The column *c_db_servertype* can be "ORACLE" or "SQLSERVER".
12. The column *c_db_nicename* is a string that can be used in client software to describe a data source.

13. The column *c_db_tooltip* contains a longer (hierarchical) representation of the nicename.

To restate, many cells need to access some kind of persistent storage, and these cells will organize their persistent storage so that it is self contained and can be apportioned in a way consistent with the project-based requirements of i2b2 described above. To that end, a table exists in many cells to make the decision of what persistent storage location to which a specific user will be directed, depending on the project and domain to which they are associated.

4.2 Schemas within the workplace data source

The following schemas provide data used by the WORK system:

4.2.1 Table_Access table

The **workplace_access schema** expands upon the WDO, containing security information for the root level work nodes. It is used to obtain a list of "Workplace" tables within the **Persistent Storage Location (PSL)**. Each Workplace table within the PSL is represented by a single row in this table. The primary identifier of each Workplace table in the **workplace_access** table is in the *c_table_cd* column. All messages that need to point to a specific workplace table will use this identifier, for example in the <Key> element of many messages where this identifier is represented as *c_table_name*\ (see messaging specification). Within the *c_table_name* column is the actual name in the PSL of the workplace table. The *c_protected_access* column designates whether one must have the protected access role (data_prot) to obtain data form this table. The other columns are defined in a similar manner to the columns of the Workplace table, with the following special notes:

The *c_hlevel* is almost always 0 in this table.

WORKPLACE_ACCESS		
PK	**c_index**	**VARCHAR(255)**
	c_table_cd	**VARCHAR(255)**
	c_table_name	**VARCHAR(255)**
	c_protected_access	CHAR(1)
	c_hlevel	**INT**

	c_name	**VARCHAR(255)**
	c_user_id	**VARCHAR(255)**
	c_group_id	**VARCHAR(255)**
	c_share_id	VARCHAR(255)
	c_visualattributes	**CHAR(3)**
	c_parent_index	VARCHAR(255)
	c_tooltip	VARCHAR(255)
	c_entry_date	DATE
	c_change_date	DATE
	c_status_cd	CHAR(1)

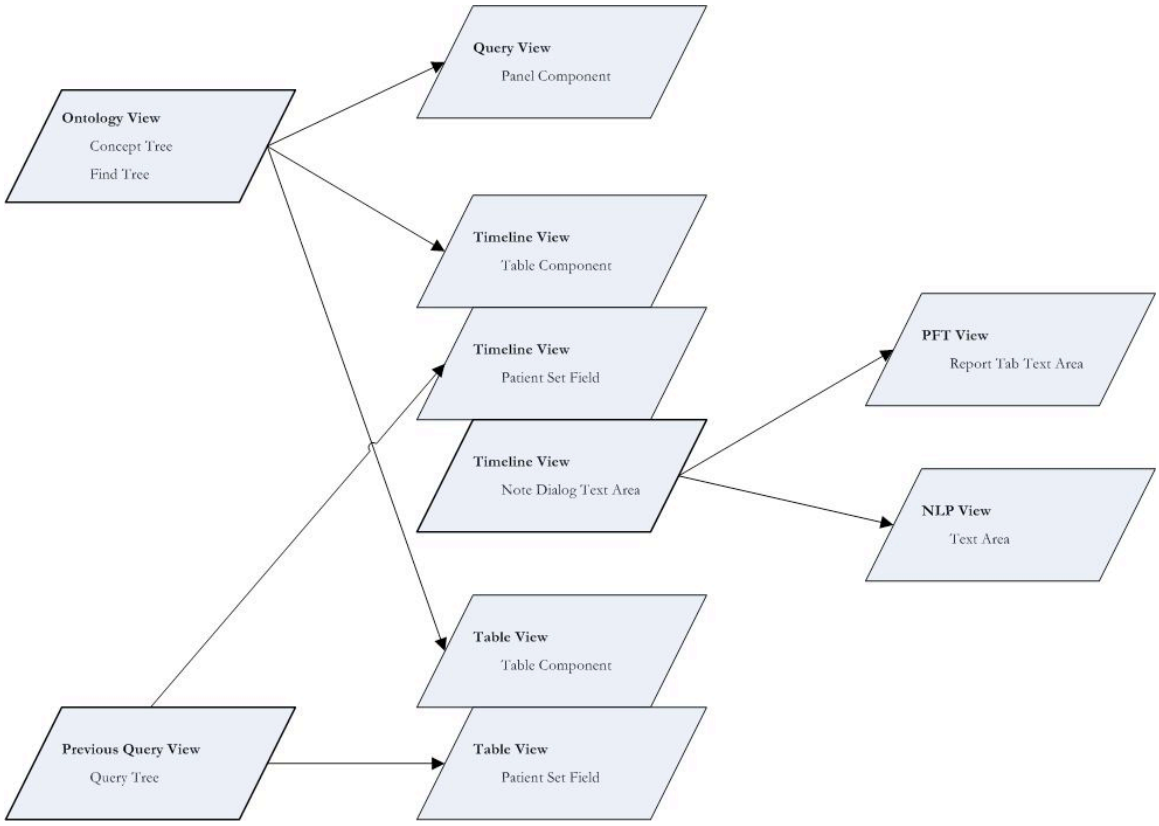
4.2.2 Workplace table

The **Workplace** table encapsulates workplace storage items used in the data repository. A folder is a work data object encapsulation of a row from the Workplace table. It is the primary object used to pass workplace items to the requesting client. A PSL may contain numerous Workplace tables, each with a name that indicates the domain that the workplace storage items contained within represents.

WORKPLACE		
PK	**c_index**	**VARCHAR(255)**
	c_name	**VARCHAR(255)**
	c_user_id	**VARCHAR(255)**
	c_group_id	**VARCHAR(255)**
	c_share_id	VARCHAR(255)
	c_visualattributes	**CHAR(3)**
	c_parent_index	**VARCHAR(255)**
	c_tooltip	VARCHAR(255)
	c_protected_access	CHAR(1)
	c_work_xml	CLOB
	c_work_xml_schema	CLOB
	c_work_xml_i2b2_type	VARCHAR(255)
	c_entry_date	DATE
	c_change_date	DATE
	c_status_cd	CHAR(1)

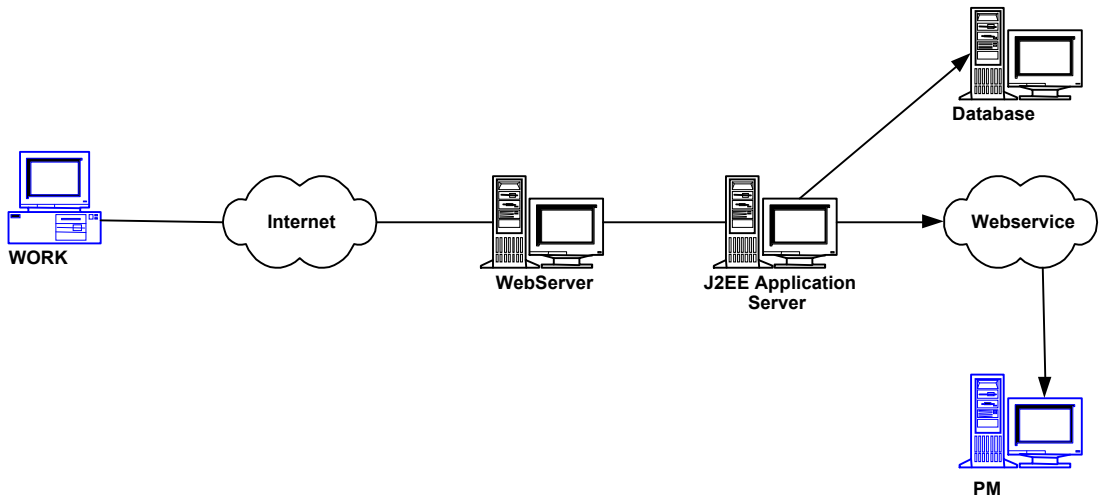
5. DRAG AND DROP ACTIONS

5.1 Current

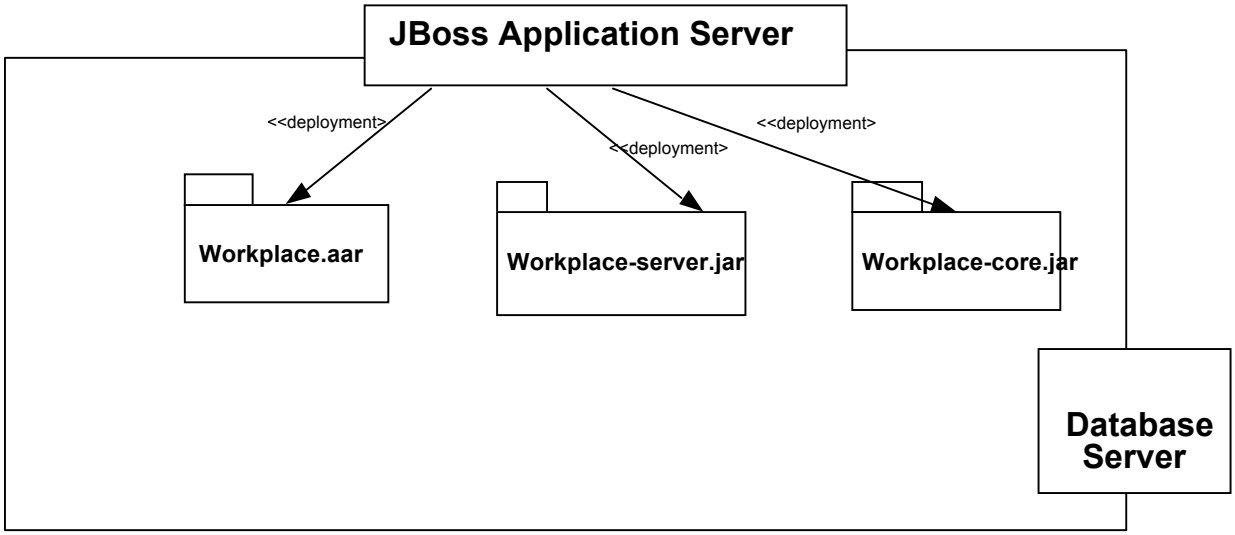


6. DEPLOYMENT VIEW

6.1 Global Overview



6.2 Detailed Deployment Model



REFERENCES

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R. and Stafford, J., (2003). Documenting Software architectures – Views and Beyond. Addison Wesley, Boston, MA.

The “4+1” view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

Object Management Group UML 2.0 Specification - http://www.omg.org/technology/documents/formal/uml.htm

i2b2 (Informatics for Integrating Biology and the Bedside) https://www.i2b2.org/resrcs/hive.html