

The information contained in this document is the property of Infinite Computer Solutions. Except as specifically authorized in writing by Infinite Computer Solutions, the holder of this document shall: (1) keep all information contained herein confidential and shall protect same in whole or in part from disclosure and dissemination to all third parties and, (2) use same for operating and maintenance purposes only.

Version: 1.4.0

Date: Nov 24, 2021

**Data Warehouse Technical**

**Specifications API Guide**

###### Edition Notice

**Note**: This edition applies to Infinite BrassRing and to all subsequent releases and modifications until otherwise indication in new editions.

Licensed Materials - Property of Infinite Talent.

Table of Contents

[Chapter 1: Introduction 3](#_Toc88563667)

[1.1 What is a Data-warehouse 3](#_Toc88563668)

[1.2 Purpose 4](#_Toc88563669)

[1.3 Audience 4](#_Toc88563670)

[Chapter 2: BrassRing’s Data-warehouse 5](#_Toc88563671)

[2.1 Overview 5](#_Toc88563672)

[2.2 Communication Methodology 5](#_Toc88563673)

[2.3 Rules for the Data-warehouse Integration 5](#_Toc88563674)

[Chapter 3: Data-warehouse files 6](#_Toc88563675)

[3.1 Standard data-warehouse files 6](#_Toc88563676)

[3.2 Business Rules on handling Delta files 7](#_Toc88563677)

[3.3 Entity Relationship Diagram (ERD) 10](#_Toc88563678)

[3.4 Sample data-warehouse Files 11](#_Toc88563679)

[3.5 Data-mapping document 11](#_Toc88563680)

[3.6 Entity Relationship Diagram (ERD) attachment 11](#_Toc88563681)

[3.7 DDL Scripts for SQL Server databases only (if needed) 11](#_Toc88563682)

[Chapter 4: Security 12](#_Toc88563683)

[4.1 Standard Implementation Security 12](#_Toc88563684)

[4.1.1 Secure FTP Server (FTPS and SFTP) 12](#_Toc88563685)

[4.1.2 PGP Encryption 12](#_Toc88563686)

[Chapter 5: Project tasks during implementation 13](#_Toc88563687)

[Notices 14](#_Toc88563688)

1. Introduction
   1. What is a Data-warehouse?

In computing, a data warehouse or enterprise data warehouse (DW, DWH, or EDW) is a database used for reporting and data analysis. It is a central repository of data which is created by integrating data from one or more disparate sources. Data warehouses store current as well as historical data.

BrassRing data-warehousing files are an extract of most of the information collected within BrassRing to a BrassRing customer for additional reporting, data analysis and data-warehousing purpose. The historical snapshots stored in the Data Warehouse are periodically refreshed with data from the operational databases.

**Advantages:**

The data warehouse addresses these factors and provides many advantages including:

* Secondary repository of the data in addition to the operational system.
* Additional documentation of the data
* Providing a place to combine related data from separate sources
* Empowering end-users to perform any level of ad-hoc queries or reports without impacting the performance of the operational systems
  1. Purpose

The purpose of this document is as follows:

* To present high-level architecture and design of BrassRing Data-warehouse and how BrassRing implements it to meet its customers’ needs for automated Foundation data management.
* To describe suggested high-level workflow processes that can be supported between BrassRing and any external, third-party system.
* To describe the steps required to implement Data-warehousing
  1. Audience
* Client Decision Makers, HRIS Implementation Teams, Internal IT Teams, Systems Integrators and Support Teams.
* BrassRing Engineering Services Team, Support Team, and Technical Services Group.

1. BrassRing’s Data-warehouse
   1. Overview

BrassRing product is the Applicant Tracking System (ATS) which Recruiters and Hiring Manager log on to create Requisitions and manage the candidates in the pipeline and move them through the recruitment process.

BrassRing houses information related to everything around recruitment which includes current and historical information on requisitions, candidates, HR Statuses, codes and configurations in the system.

Data-warehouse is the process to export all the above information over to the Customer as a nightly process in a structured way in the form of delimited files that can be interpreted as tables.

* 1. Communication Methodology

FTP process is the data transfer vehicle for the delivery of the data-warehousing files. The FTP server can be a BrassRing hosted server or a Customer hosted server depending on the customer’s preference. The FTP server protocols that are supported are standard FTP, SFTP or FTPS with a username and password authentication.

There are several options on how the files can be delivered.

1. TXT files
2. PGP encrypted
3. ZIP file
4. ZIP and then PGP encrypted
   1. Rules for the Data-warehouse Integration

Below are a few rules to around the DW process:

* The DW files are a set of standard files as determined in the data mapping spreadsheet.
* The terminology “tables” and “files” are used interchangeable throughout the document as per the context of the sentence.
* The process runs as a nightly batch scheduled between 1 AM US EST – 8 AM US EST.
* The scheduling days are extremely customizable depending on customer’s requirements.
* BrassRing maintenance day is always 3rd Saturday of every month so we recommend not to run the scheduler on Saturdays.
* The first time the DW files are generated is a full load from Day 1 to current date and thereafter the process only generates deltas. The rules on how the Delta file need to be handled is listed below in 3.26.
* Field mappings are outlined in the Data Mapping document.
* The Entity Relationship Diagram (ERD) explains how the tables are joined with each other and the relationships with primary keys and foreign keys in the tables.

1. Data-warehouse files
   1. Standard data-warehouse files

The data-warehouse process generates 34 files that contain BrassRing data relevant for data-warehousing purposes.

These 34 files are namely:

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **CANDIDATE** | All Candidates | Master |
| **2** | **EDUCATION** | Education data from candidates’ profile. | Transactional |
| **3** | **EXPERIENCE** | Experience data from candidates’ profile. | Transactional |
| **4** | **STACKING** | Tracks candidate records that have been manually stacked. | Transactional |
| **5** | **ELINK** | Contains information on eLinks sent, responded, viewed etc. Useful for eLink tracking reports. | Transactional |
| **6** | **CODESDETAIL** | Codes that came along with a candidate submission | Transactional |
| **7** | **HRSTATUS\_MAIN** | Roll up HRStatus table listing out current HRStatus at a requisition level. | Transactional |
| **8** | **HRSTATUS\_HISTORY** | Historical data on HRSTATUS\_MAIN | Transactional |
| **9** | **FORM\_INSTANCE** | Custom candidate forms attached to a candidate and Requisition Addendum Forms | Transactional |
| **10** | **RESPONSE** | Response data for fields on candidate forms and Req addendum forms. | Transactional |
| **11** | **CANDIDATETYPES** | Master file for the Candidate Types | Master |
| **12** | **CANDIDATETYPEHISTORY** | Contains the historical details on the Candidate Type for a Candidate | Transactional |
| **13** | **REFERRALS** | Contains the referral information | Transactional |
| **14** | **FORM\_TYPE** | Listing of all custom forms (both requisition and candidate) | Master |
| **15** | **CODESMASTER** | List of source codes, job codes and other code types | Master |
| **16** | **QUESTION\_TYPE** | Listing of all fields present on a form | Master |
| **17** | **OPTIONS** | Custom list of choices setup for fields on a form | Master |
| **18** | **TG\_SITES** | Listing of all Talent Gateway Sites | Master |
| **19** | **GRID\_ROWS** | Listing of Grid Row Labels. Grid question responses take the form of x\_y where x is the rowID. | Master |
| **20** | **GRID\_COLUMNS** | Listing of Grid Column Labels. Grid question responses take the form of x\_y where y is the columnID. | Master |
| **21** | **REQUISITIONS** | Contains all requisitions. | Transactional |
| **22** | **FOLDER** | Contains folder information for requisition, working folders and inbox. | Transactional |
| **23** | **REQ\_POSTING\_ACTIVITY** | Information on Requisition Posting to Talent-Gateways. | Transactional |
| **24** | **REQ\_POSTING\_HISTORY** | Historical information on posting to Talent-Gateways. Future dated postings are not included in this file. | Transactional |
| **25** | **REQ\_STATUS\_HISTORY** | History table identifying who modified a requisition and its status changes. | Transactional |
| **26** | **REQUISITION\_APPROVALS** | Information on Approvals setup on a requisition. | Transactional |
| **27** | **REQUISITIONRESPONSE** | Responses to custom questions setup on requisition (form) and Req subsidiary forms. RequisitionResponse data will always be sent when Requisition data is sent. | Transactional |
| **28** | **REQUISITIONTEAM** | Track Req Team Members: ReqTeam Members will be sent when Requisitions Data is sent. \*\*\*Delete ReqTeam\_Members for FolderID and reinsert.\*\*\* | Transactional |
| **29** | **SEARCH\_LOG\_MASTER** | OFCCP Data. Log of every search on candidate. (Log File for Audit Purposes) | Master |
| **30** | **SEARCH\_LOG\_DETAILS** | OFCCP Data. Log of every search on candidate. (Log File for Audit Purposes) | Transactional |
| **31** | **USERS** | All Users of the system. | Master |
| **32** | **USER\_TYPE** | Category of user; used for Security. | Transactional |
| **33** | **USER\_APPROVAL** | Association of a user with an Approval Level. | Transactional |
| **34** | **USER\_ORG\_GROUP** | Association of a user with an OrgGroup. | Transactional |
| **35** | **FORMS\_DELETE\_HISTORY** | Contains information of candidate forms deleted. | Transactional |

* 1. Business Rules on handling Delta files

One general note while processing the data-warehouse files is that the Master files must be loaded first before the transactional files are loaded.

The sequence for the Master files is below:

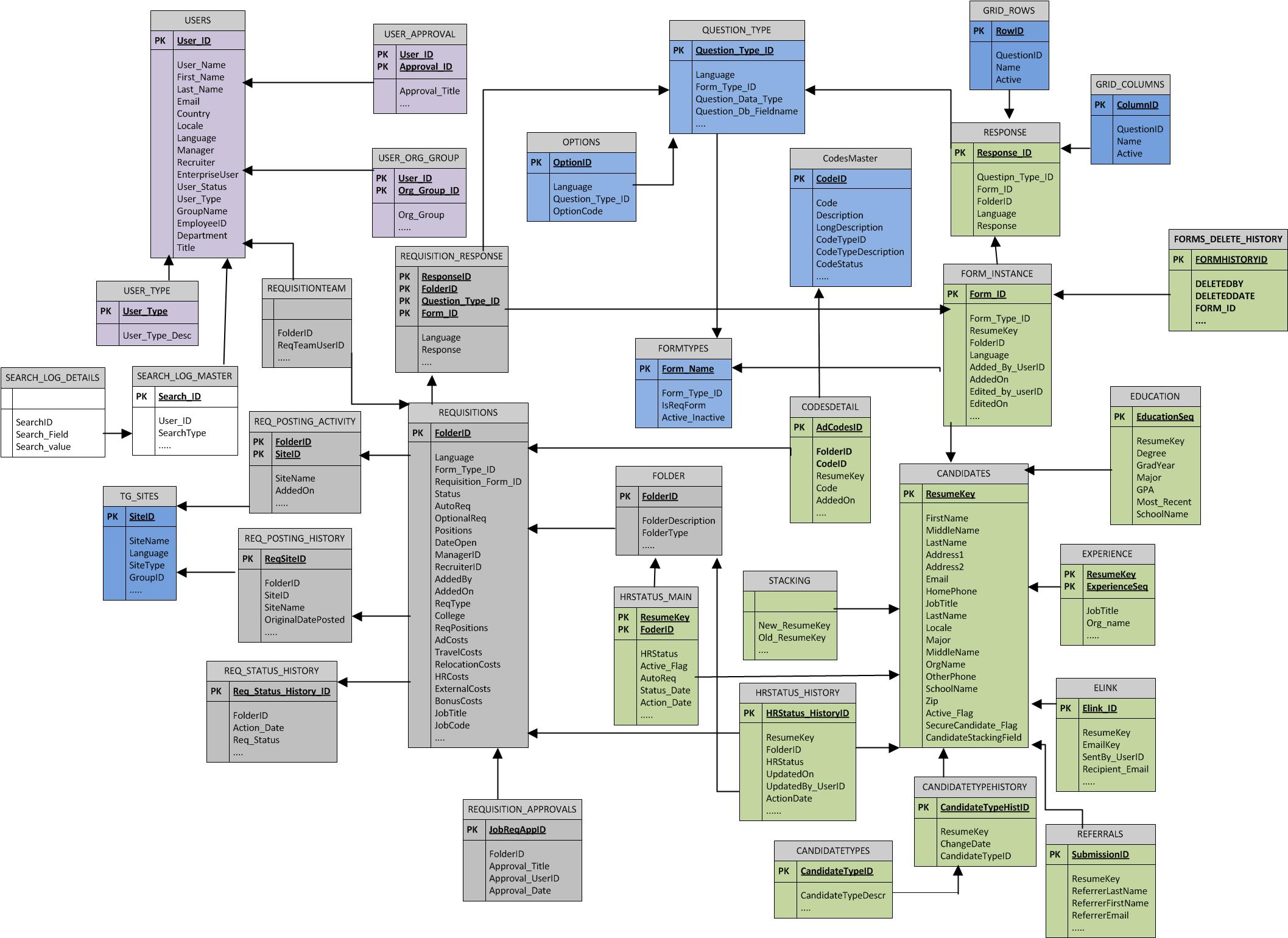
1. CANDIDATE
2. USERS
3. FORMTYPE
4. CODESMASTER
5. QUESTION\_TYPE
6. OPTIONS
7. TG\_SITES
8. GRID\_ROWS
9. GRID\_COLUMNS
10. REQUISITIONS
11. SEARCH\_LOG\_MASTER

Below are important additional business runes that need to be built in on the Client side when processing the files to ensure the integrity and accuracy of the data on the delta process.

|  |  |  |
| --- | --- | --- |
| **1** | CANDIDATE | Check If RESUMEKEY exists and CANDIDATE\_TYPE is Erased. If Yes, then DELETE. If RESUMEKEY exists and CANDIDATE\_TYPE <> Erased, then UPDATE else INSERT |
| **2** | EDUCATION | The Education data is always provided when CANDIDATE data is sent. When it is determined that the record in the CANDIDATE file is an update then DELETE all EDUCATION for the specific RESUMEKEY and re-INSERT values again coming from this file |
| **3** | EXPERIENCE | The Experience data is always provided when CANDIDATE data is sent. When it is determined that the record in the CANDIDATE file is an update then DELETE all EXPERIENCE for the specific RESUMEKEY and re-INSERT values again coming from this file |
| **4** | STACKING | UPDATE Old ResumeKey with New ResumeKey in all tables except CANDIDATE, EDUCATION, EXPERIENCE, HRSTATUS\_MAIN, and HRSTATUS\_HISTORY.  DELETE or flag as inactive all records referencing the Old Resumekey from the CANDIDATE, EDUCATION, and EXPERIENCE tables.  If both Old and New Resumekeys exist in HRSTATUS\_MAIN or HRSTATUS\_HISTORY for a given FolderID  then DELETE all records with Old Resumekey and FolderID from HRSTATUS\_MAIN and HRSTATUS\_HISTORY tables  Else UPDATE Old Resumekey with New Resumekey for all other records in HRSTATUS\_MAIN and HRSTATUS\_HISTORY tables. |
| **5** | ELINK | Check If ELINK\_ID exists. If Yes then UPDATE else INSERT |
| **6** | CODESDETAIL | Check If ADCODESID exists. If Yes then UPDATE else INSERT |
| **7** | HRSTATUS\_MAIN | Check if RESUMEKEY and FOLDERID combination exists. If Yes then UPDATE else INSERT |
| **8** | HRSTATUS\_HISTORY | Check if RESUMEKEY and FOLDERID combination exists. If Yes then DELETE existing records with that combination and re-INSERT |
| **9** | FORM\_INSTANCE | Check If FORM\_ID exists. If Yes then UPDATE else INSERT |
| **10** | RESPONSE | This data is provided every time a Candidate Form or Req Addendum Form is updated. If form id in delta file exists, then delete all records from the RESPONSE table with that FORM\_ID and re-INSERT values coming from this file. Insert records for FORM\_IDs. |
| **11** | CANDIDATETYPES | Check If CANDIDATETYPEID exists. If Yes, then UPDATE else INSERT |
| **12** | CANDIDATETYPEHISTORY | Check If CANDIDATETYPEHISTORYID exists. If Yes, then UPDATE else INSERT |
| **13** | REFERRALS | Check If SUBMISSIONID exists. If Yes, then UPDATE else INSERT |
| **14** | FORM\_TYPE | Check If FORM\_TYPE\_ID exists. If Yes, then UPDATE else INSERT |
| **15** | CODESMASTER | Check If CODEID exists. If Yes then UPDATE else INSERT |
| **16** | QUESTION\_TYPE | Check If QUESTION\_TYPE\_ID exists. If Yes, then UPDATE else INSERT |
| **17** | OPTIONS | Check If OPTIONID exists. If Yes, then UPDATE else INSERT |
| **18** | TG\_SITES | Always a full file. DELETE all TG\_SITES and re-INSERT values again coming from this file |
| **19** | GRID\_ROWS | Always a full file. DELETE all GRID\_ROWS and re-INSERT values again coming from this file |
| **20** | GRID\_COLUMNS | Always a full file. DELETE all GRID\_COLUMNS and re-INSERT values again coming from this file |
| **21** | REQUISITIONS | Check If FOLDERID exists. If Yes, then UPDATE else INSERT |
| **22** | FOLDER | Check If FOLDERID exists. If Yes, then UPDATE else INSERT |
| **23** | REQ\_POSTING\_ACTIVITY | Check if FOLDERID and SITEID combination exists. If Yes, then UPDATE else INSERT |
| **24** | REQ\_POSTING\_HISTORY | Check If REQSITEID exists. If Yes, then UPDATE else INSERT |
| **25** | REQ\_STATUS\_HISTORY | Check If REQ\_STATUS\_HISTORY\_ID exists. If Yes, then UPDATE else INSERT |
| **26** | REQUISITION\_APPROVALS | Check If JOBREQAPPID exists. If Yes then UPDATE else INSERT |
| **27** | REQUISITIONRESPONSE | This data is provided every time a Requisition or Req Subsidiary Form is updated. If FORM\_ID in delta file exists,, then delete all records from the REQUISITIONRESPONSE table with that FORM\_ID and re-INSERT values coming from this file. Insert records for new FORM\_IDs. |
| **28** | REQUISITIONTEAM | The Req Team data is always provided when REQUISITIONS data is sent. When it is determined that the record in the REQUISITIONS file is an update then DELETE all REQUISITIONTEAM for the specific FOLDERID and re-INSERT values again coming from this file |
| **29** | SEARCH\_LOG\_DETAILS | Always an INSERT. No check required |
| **30** | SEARCH\_LOG\_MASTER | Check If SEARCHID exists. If Yes, then UPDATE else INSERT |
| **31** | USERS | Check If USER\_ID exists. If Yes then UPDATE else INSERT |
| **32** | USER\_TYPE | Always a full file. DELETE all USER\_TYPE and re-INSERT values again coming from this file |
| **33** | USER\_APPROVAL | The User Approval data is always provided when USERS data is sent. When it is determined that the record in the USERS file is an update then DELETE all USER\_APPROVAL for the specific USER\_ID and re-INSERT values again coming from this file |
| **34** | USER\_ORG\_GROUP | The User Approval data is always provided when USERS data is sent. When it is determined that the record in the USERS file is an update then DELETE all USER\_ORG\_GROUP for the specific USER\_ID and re-INSERT values again coming from this file |
| **35** | FORMS\_DELETE\_HISTORY | This data is provided for deleted candidate forms. DELETE from FORM\_INSTANCE and RESPONSE for the FORM\_ID |

* 1. Entity Relationship Diagram (ERD)

Entity Relationship Diagrams (ERD) is a graphical representation of entities and their possible relationships with each other.



* 1. Sample data-warehouse Files

Attached below are the sample physical files that are generated as part of this process.



**Note:** The data in sample files is generic data and doesn’t reflect any client specific configuration.

* 1. Data-mapping document

Attached below is the data-mapping document. The data mapping document outlines the fields within each file and describes the files in greater details. The datatype, size and field descriptions can also be found here.



* 1. Entity Relationship Diagram (ERD) attachment



* 1. DDL Scripts for SQL Server databases only (if needed)

Attached below is the Data Dictionary Scripts (DDL) for creating the tables on a SQL Server database. For other databases such as Oracle and DB2, client will have to generate their own scripts or modify the provided DDL to suit the RDBMS.



**Note:** The script attached is only for informational purpose and will not guarantee all the relationships between entities.

1. Security

Several security protocols and measures have been implemented to ensure security on the data transmission, message integrity, and message confidentiality. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

* 1. Standard Implementation Security
     1. Secure FTP Server (FTPS and SFTP)

Message integrity is ensured on the XML data. Unique credentials within the XML act as authentication information for a transaction to be consumed. These credentials in the Id, Credential and Manifest nodes act as unique value for a given implementation.

Data-warehouse files are delivered via an FTP process. The data security of the files is secured by the nature of the FTP server. BrassRing supports FTP, SFTP and FTPS mechanism for delivering the files.

FTP authentication is via a username and password combination.

*Note: Currently, FTPS authentication using server-side public key authentication certificates and client-side authorization certificates is not available.*

* + 1. PGP Encryption

PGP is a data encryption and decryption technology that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting and decrypting texts, e-mails, files, directories and whole disk partitions to increase the security of e-mail communications.

PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms. Each public key is bound to a username and/or an e-mail address.

For PGP encryption, customer provides the PGP public key to BrassRing in an .ASC format. BrassRing uses this PGP key for encrypting the data-warehouse files.

1. Project tasks during implementation

Below is the list of tasks that typically is executed during a data-warehouse implementation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Task Name** | **Duration** | **Predecessors** | **Resources** |
| 1 | Kick off Meeting | 1 day |  |  |
| 2 | Touch base, confirm scope, discuss timelines and expectation | 1 day |  | BrassRing, Client |
| 3 | File Transfer Mechanisms | 4 days |  |  |
| 4 | Exchange PGP keys if necessary and check connectivity with FTP site | 1 day | 1 | BrassRing, Client |
| 5 | Create FTP server account (Client side or BrassRing side) | 2 days | 4 |  |
| 6 | Exchange FTP information and verify connectivity | 1 day | 5 |  |
| 7 | Development Efforts | 20 days |  |  |
| 8 | Review of Data Mapping and Data Diagram | 5 days | 6 | BrassRing/Client |
| 9 | BrassRing development for data warehousing | 15 days | 8 |  |
| 10 | Client development for data warehousing | 15 days | 8 | Client |
| 11 | First Dry Run (Full Load of all data) - Level 1 Testing | 24 days |  |  |
| 12 | Generate full load of all data | 5 days | 10 | BrassRing |
| 13 | Review of Full load data files | 4 days | 12 | BrassRing |
| 14 | Client loads full data into Client’s reporting application | 2 days | 12 | Client |
| 15 | Client generates new data in the BrassRing system | 2 days | 14 | Client |
| 16 | Schedule delta load generation at a Frequency | 1 day |  | BrassRing |
| 17 | Review of First delta load data files | 4 days | 16 | BrassRing |
| 18 | Client loads delta feeds into Client’s reporting application | 1 days | 17 | Client |
| 19 | Client reviews data and validates delta generation | 3 days | 18 | Client |
| 20 | Change/Fix items identified during dry run | 2 days | 19 | BrassRing |
| 21 | Second Dry Run (Full Load of all data) - Level 2 Testing | 19 days |  |  |
| 22 | Generate full load of all data | 3 days | 20 | BrassRing |
| 23 | Review of Full load data files | 4 days | 22 | BrassRing |
| 24 | Client loads full data into Client’s reporting application | 2 days | 23 | Client |
| 25 | Schedule delta load generation at a Frequency | 1 day | 24 | BrassRing |
| 26 | Review of first delta data files | 4 days | 25 | BrassRing |
| 27 | Client loads delta feeds into Client’s reporting application | 1 days | 26 | Client |
| 28 | Client reviews data and validates delta generation | 2 days | 27 | Client |
| 29 | Change/Fix items identified during dry run | 2 days | 28 | BrassRing |
| 30 | Client provides UAT Sign off | 5 days | 29 | Client |
| 31 | Production Move and Go Live! | 14 days |  |  |
| 32 | Generate full load of all data | 5 days | 30 | BrassRing |
| 33 | Review of Full load data files | 4 days | 32 | BrassRing |
| 34 | Schedule delta load generation | 1 day | 31 | BrassRing |
| 35 | Review of first delta data files | 4 days | 34 | BrassRing |

***Typical project duration is 87 days***