CHAKRA – Technical Whitepaper

ANSHINSOFT

Anshinsoft Corp.
5201 Great America Parkway, Suite 320
Santa Clara, CA 95054. USA.
Phone: +1-408-730-2634

www.anshinsoft.com

TABLE OF CONTENTS

1 About CHAKRA

CHAKRA is a comprehensive, modular and customizable family of solutions for automating the entire post-execution trade life-cycle management, enabling Straight through Processing (STP) in the back office operations. It provides out-of-the-box application components covering all major processes, starting from trade capture & enhancement through all the way to accounting & reporting. It has been built on state-of-the-art Java EE platform, based on a configurable, message-driven architecture, offering flexibility, scalability and extensibility, to make the solution adaptable for different kind of businesses (brokers, custodians, asset managers, hedge funds, etc.) in any capital markets around the globe.

The CHAKRA family of solutions includes the following separately deployable yet seamlessly interoperable components:

CHAKRA Back Office

Comprehensive, customizable back office solution that enables brokers and custodians to achieve intra-firm and inter-firm STP in post trade lifecycle management across markets, asset classes and currencies.

CHAKRA Corporate Action Management

End-to-end life cycle management of Corporate Actions from announcement capture through balance maintenance, notifications, entitlement generation, payments and taxation to fail tracking.

CHAKRA Reconciliation

User configurable reconciliation engine that handles multiple data sources to "reconcile anything with anything".

CHAKRA Dashboard

Personalizable dashboard that will pull in critical data from back office processes from various systems and present them in a single auto-refreshing view with configurable alerts.

2 System Architecture

This section contains a high-level overview of the system architecture and its benefits, followed by a section that outlines the major architectural components with an implementation overview.

2.1 Overview

CHAKRA is built on Java EE technology stack and uses industry standard architectural components for seamless interoperability.

The following diagram depicts the technical architecture of CHAKRA:

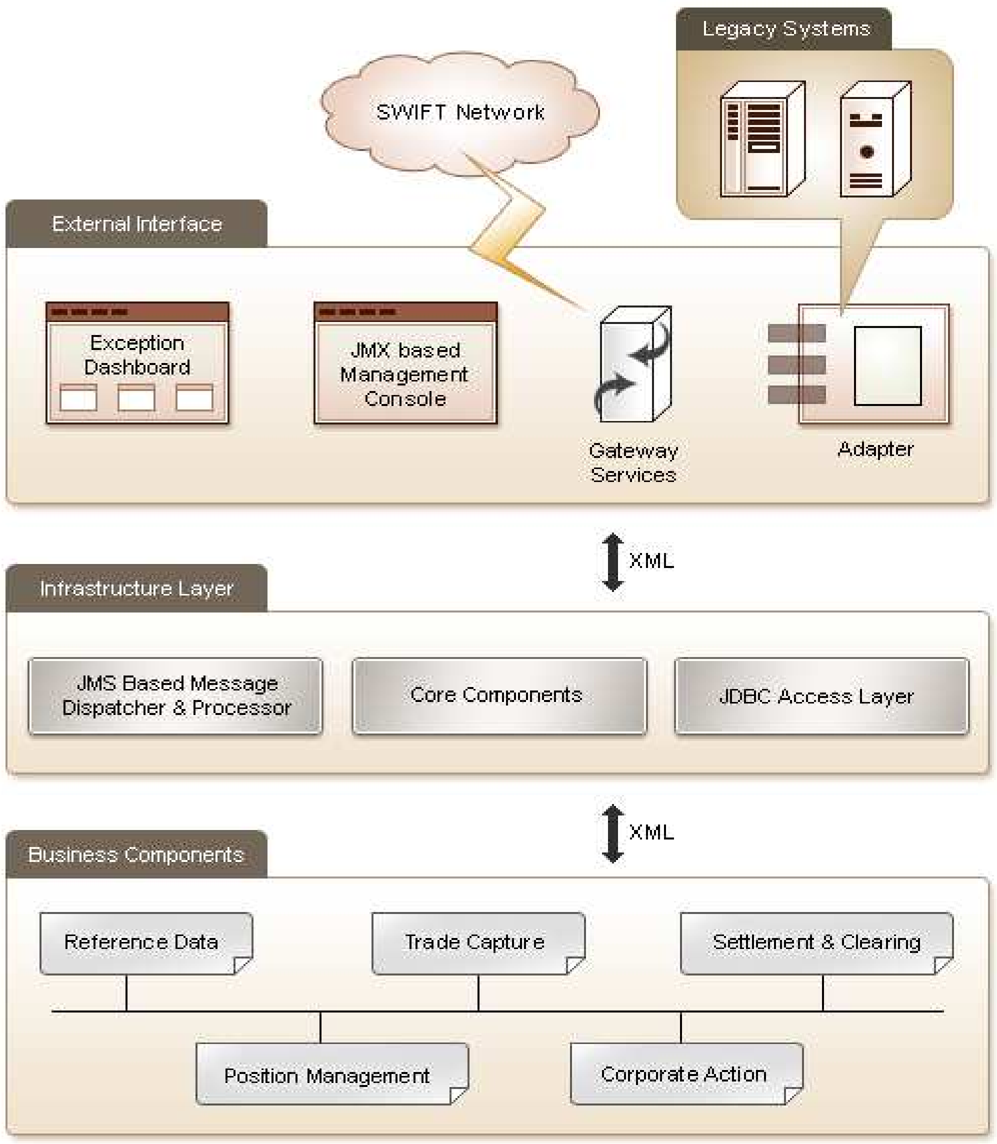

Figure 1 - System Architecture

As seen in the diagram above, CHAKRA is designed with multiple interrelated components, each implementing one part of the business process, interacting with each other through XML messages.

- The infrastructure component manages all resources (database access, message queues, etc.).
- All business components is managed by a Central Monitor, which polls component specific message queues for incoming XML messages, process them and finally write output messages in the designated output message queues of that component.
- The Message Router service will route messages across various components based on a rule base that is configurable.
- Multi-tier architecture – the back-end database ensures data integrity; the message based middleware ensures guaranteed delivery of data across modules and the browser based client ensures easy availability & manageability of the application.

2.2 Component View

The following diagram illustrates a component view of CHAKRA's artifacts. All components are independent modules that interact with other components through a message router.

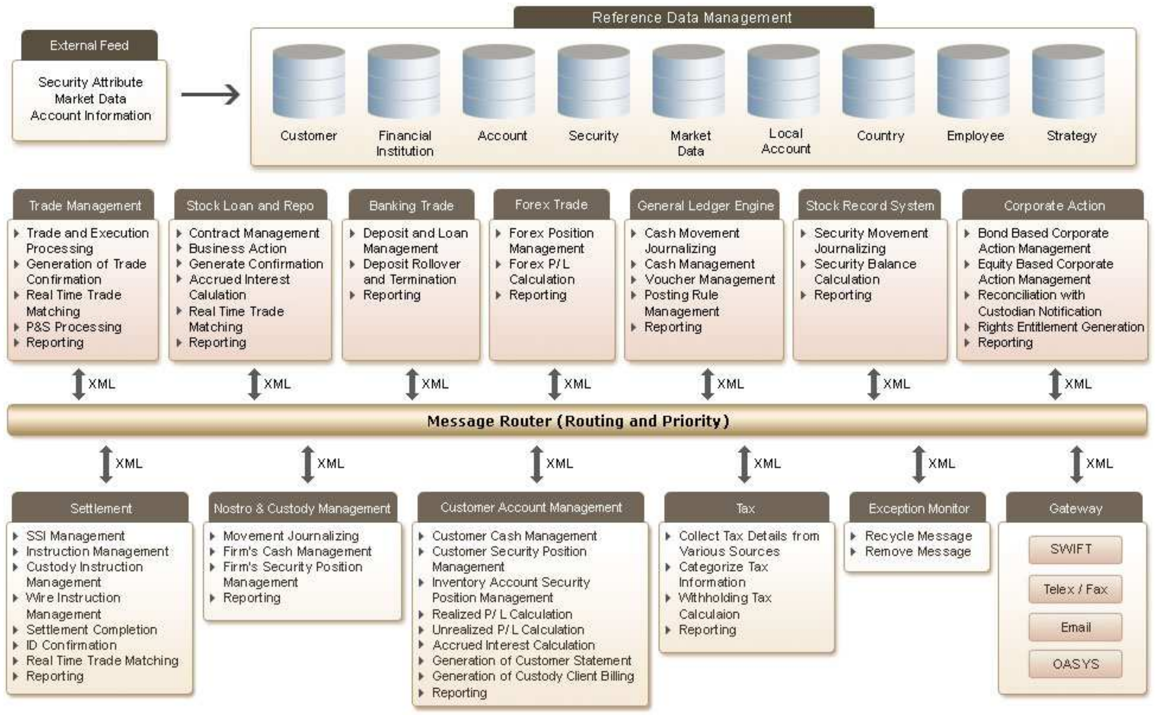
Reference Data Management
External Feed
Security Attribute
Market Data
Account Information
Customer
Financial Institution
Account
Security
Market Data
Local Account
Country
Employee
Strategy
Trade Management
Trade and Execution Processing
Generation of Trade Confirmation
Real Time Trade Matching
P&S Processing
Reporting
Stock Loan and Repo
Contract Management
Business Action
Generate Confirmation
Accrued Interest Calulation
Real Time Trade Matching
Reporting
Banking Trade
Deposit and Loan Management
Deposit Rollover and Termination
Reporting
Forex Trade
Forex Position Management
Forex P/L Calculation
Reporting
General Ledger Engine
Cash Movement Journalizing
Cash Management
Voucher Management
Posting Rule Management
Reporting
Stock Record System
Security Movement Journalizing
Security Balance Calculation
Reporting
Corporate Action
Bond Based Corporate Action Management
Equity Based Corporate Action Management
Reconciliation with Custodian Notification
Rights Entitlement Generation
Reporting
XML
Message Router (Routing and Priority)
Settlement
SSI Management
Instruction Management
Custody Instruction Management
Wire Instruction Management
Settlement Completion
ID Confirmation
Real Time Trade Matching
Reporting
Nostro & Custody Management
Movement Journalizing
Firm's Cash Management
Firm's Security Position Management
Reporting
Customer Account Management
Customer Cash Management
Customer Security Position Management
Inventory Account Security Position Management
Realized P/L Calculation
Unrealized P/L Calculation
Accrued Interest Calculation
Generation of Customer Statement
Generation of Custody Client Billing
Reporting
Tax
Collect Tax Details from Various Sources
Categorize Tax Information
Withholding Tax Calculaion
Reporting
Exception Monitor
Recycle Message
Remove Message
Gateway
SWIFT
Telex / Fax
Email
OASYS

Figure 2 - Component View

Every component works on a set of input queues, output queues and error queues. The following is the sequence of an activity within a module:

- A component (e.g. Trade) receives messages in its input queue. The message may be a Trade message from the middle office for processing into the back office system.
- The Trade component processes the message, updates all relevant database entities and generates another set of messages that needs to be processed by the downstream components (e.g. Settlement, Accounting etc.). But instead of directly interacting with the other components, it delivers the appropriate messages to its output queue.
- The Trade component does not have knowledge of any specifics of the downstream components and leaves it to the message router to handle downstream transport. The router looks up the rule base, determines the set of downstream components that needs the message, and transports the messages to their respective input queues.

Being loosely coupled through a message router, the architecture ensures reliability and scalability. It does not have a single-point-of-failure, as it is asynchronous in nature.

2.3 Architecture Highlights

Some of the main features of CHAKRA architecture are:

- **Loose coupling:** The various layers of CHAKRA are loosely coupled and interact with each other through messages. This ensures proper separation of concerns and makes the solution easy to manage, monitor and maintain.
- **XML based messaging:** CHAKRA uses XML for all internal message processing. This makes the system easy to integrate with other applications.
- **Message Driven Service Layer**: CHAKRA's service modules run on a message oriented backbone that processes messages asynchronously from their individual queues. This makes the system more resilient from failures and makes the architecture horizontally scalable.
- **Rule based configuration**: All of CHAKRA's configuration parameters, market practices and SSI settings are driven by a rule engine that is externalized in a rule database. This further adds to the flexibility of the architecture.
- **Localization of External Interface**: Any exchange of data with the external world is handled through a separate component – the Gateway. This ensures localization of the external interfaces – any processing / formatting specific to the external world is handled through interface specific connectors residing at the Gateway level.
- **Exception Monitoring Tool**: Exceptions are handled through a separate service which is responsible for tracking, monitoring and the automatic handling of business exceptions. The exception monitor is rule-driven, with rules to identify actions associated with specific classes of exceptions. This also includes a user interface for manual handling of business exceptions, if necessary.

2.4 Technology Components

Following are the technology components upon which CHAKRA is architected:

Technology Stack	Java EE and Java 2 Platform SDK
Application Server	Any Java EE compatible Application Server, including: • IBM Websphere • Weblogic • JBoss • Apache Tomcat
Presentation Media	Web Browser
Presentation Layer Framework	Struts at Server Side, Adobe Flex at Client Side
Presentation Technique	• JSP/Servlet • Flash
Messaging Middleware	Any JMS compliant messaging system; has been tested on the following: • IBM Websphere MQ • ActiveMQ
Relational Databases	Oracle 10G, MySQL, H2 (in-memory database)
Grid Infrastructure	Terracotta
Domain Model	POJO Based
Persistence Framework	Custom DAO interfacing with JDBC
Business Rules	Custom Rule Engine
Transaction Management	Global Transaction managed by Custom Developed Transaction Manager
Session Management	Cookie Based
System Logging	File logging with Log4J
System Monitoring	JMX based Admin Interface
Resource Configuration and Lookup	Application Managed
Authentication	Container Managed as per Java EE spec
Authorization	Managed by Application Roles configured in database
Reporting	Embedded Reporting Engine, XML based manual report layout configuration
Message Format	XML
Message Transformation	Custom Gateway and Adapter component
Message Routing	Custom Message Router
Message Consumer	POJO based P2P Message Consumer running as a service outside Java EE container
Service Lookup	RMI Registry
Service Architecture	POJO based - services developed on top of Custom Services Framework
Batch Processes	POJO based – developed on top of Custom Application Launcher Framework

3 Deployment Architecture

The following diagram illustrates typical deployment architecture of the system for handling moderate transaction volume (10K-20K per day). For handling higher level of volume the deployment configuration can be enhanced with higher levels of processing power and deploying additional servers in a cluster.

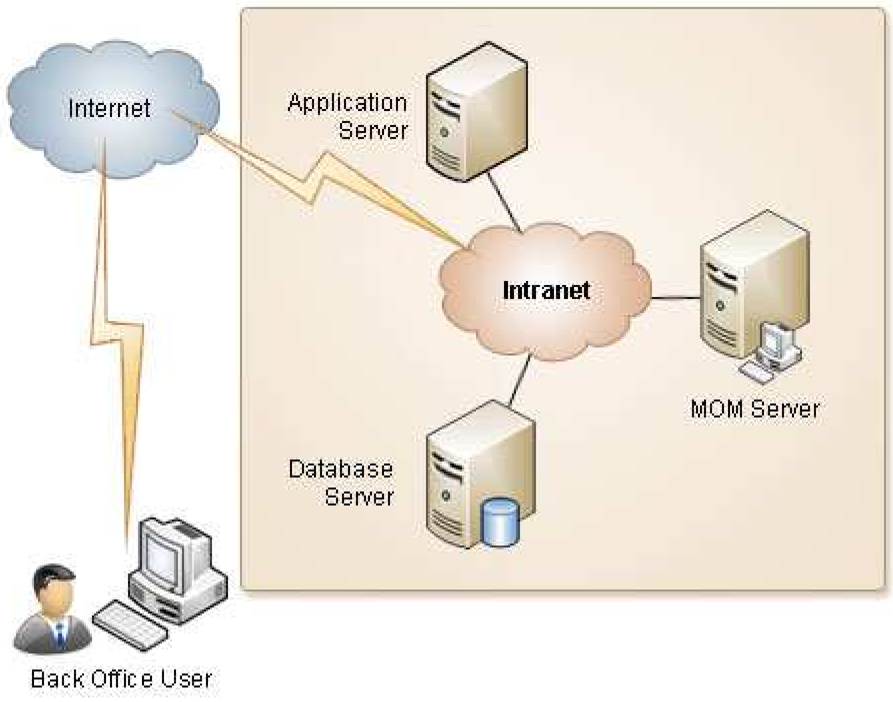

Figure 3 - Typical Deployment View

- **Application Server** – Hosts the following:
  - Java EE Application Server for the deployed back office application
  - Out-of-container Services and Batch Jobs
- **MOM Server** – Hosts the messaging middleware services
- **Database Server** – Hosts the back-end database system

3.1 Hardware and Software Requirements

3.1.1 Third party software

CHAKRA uses the following third party software:

1. **Database Server** : Oracle 10G, as well as MySQL and H2(in-memory database)
2. **Messaging Server**: ActiveMQ or IBM Websphere MQ. ActiveMQ is a lightweight server, available as Open Source software. IBM Websphere MQ is advised as a

preferred choice for more demanding production environments requiring high transaction throughput.

3. **Java EE Application Server**: Tomcat, JBoss, IBM Websphere or Weblogic. Tomcat and JBoss are both lightweight, Open Source servers and any one of them is preferred at a typical deployment.

3.1.2 Operating System

Preferred operating system for deploying and running CHAKRA is 32-bit Linux or any other compatible UNIX system.

3.1.3 Web Browser Dependency

CHAKRA web interface is browser agnostic. However, it does require Flash Player to be installed in the client machines.

3.1.4 Hardware

By its nature of being a distributed application, CHAKRA requires multiple machines for deployment and operation. The basic system configuration requires a maximum of three servers and a minimum of two servers. The system runs in a distributed environment and the servers can be located in different physical locations, if required. Depending on transaction volume, servers can be configured at different levels of processing power and additional servers may be deployed in a cluster. For a typical mid-volume scenario the following configurations may be applicable:

3.1.4.1 Database Server

Quantity required: 1

Typical Specification

Quad Core Intel Xeon CPU, with shared 2X4MB L2 cache at up to 2.66GHz

4GB PC2-5300 667MHz fully buffered DDR2 memory

4x100 GB 10k rpm SAS HDD in RAID 1+0 (ideal for Database environment)

3.1.4.2 Messaging and Application Servers

Quantity required: 2

Typical Specification

Quad Core Intel Xeon CPU, with shared 2X4MB L2 cache at up to 2.66GHz

4GB PC2-5300 667MHz fully buffered DDR2 memory

3x146 GB 15K RPM SAS HDD in RAID 5

3.2 Scalability & Failover

The deployment architecture as illustrated in the previous sections describes the basic Java EE compliant application server based model. For issues like scalability, application-failover and performance, CHAKRA can be scaled up in an environment where Database, Messaging Middleware and Java EE Application Servers are clustered by their native clustering mechanisms. Architecturally, CHAKRA is aware of these external clusters and will react properly in the event of node failures. Following external clusters are supported:

1. Oracle 10G RAC
2. Websphere MQ
3. IBM Websphere Application Server Cluster

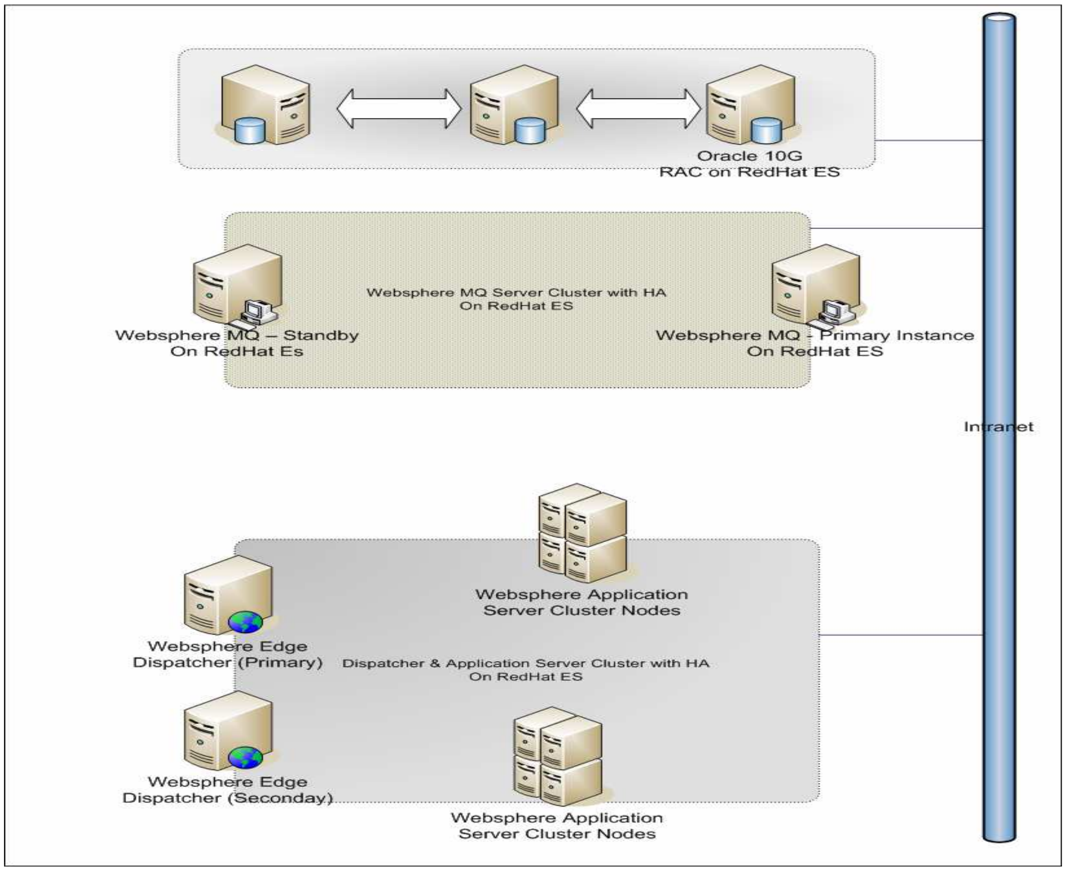

Figure 4 - Clustered Architecture

The clustering model provides additional value to the basic deployment architecture (described in the previous section) and addresses the typical concerns of a mission critical distributed application. The figure shows the distribution of the application over clusters of application servers, thereby contributing to its overall performance, availability, scalability, reliability and manageability. The following points show how these issues are managed by adding clusters and tiers to the proposed application architecture:

High Availability – The proposed architecture is HA enabled, through usage of HA software within each tier. This ensures that all essential services are restored quickly in the event of a system, component or application failure. The HA clustering can be architected either with standby mode or takeover mode. Clustered application architecture makes several independent tiers and servers appear as one tier to clients accessing the application. Ensuring that a particular service is duplicated across multiple servers guarantees high availability, since some servers are always available to service a client request. The server that actually responds to a specific request is usually determined via a load-balancing mechanism, which distributes requests to different servers.

Scalability - Application design takes into account factors such as the number of concurrent clients accessing an application, the response time for service requests, transaction throughput, and so forth. A cluster's ability to add or remove servers without altering the basic application architecture allows the application to grow with the business while continuing to leverage the initial investment in the application's design and development.

Performance - The clustered model suggested above leads to better performance since the application is able to perform under high-load conditions irrespective of the number of requests made.

Load Balancing - All application servers come bundled with an automatic load balancer which determines the server to which the user request will be routed, so that the system is always maintained in optimum load condition. The load balancer is also responsible for detecting application fail-over, when it automatically redirects the request to another server within the cluster that is capable of servicing the request.

4 Integration Architecture

CHAKRA relies on two major components for integration with systems outside the boundary of core back-office functions.

1. Adapter – For interfacing with in-house or external systems such as Front and Middle Office and General Ledger, etc.
2. Gateway – For interfacing with market intermediaries such as custodians, clearing corporations, exchanges, etc.

The CHAKRA integration layer is designed on open standards (XML) based message oriented architecture. To interface with external systems which accept messages in formats other than XML, the Gateway/Adapter layers can be extended with custom development of message-format adapters. This makes it possible to interface with any external system having proprietary message formats & protocols, provided the message format meta-data is available and can be mapped to CHAKRA's XML message schema. SWIFT message adapters are provided out of the box.

CHAKRA also provides a Java API for Data Access which enables other applications to query all transaction and reference data stored in the CHAKRA database. The standard Excel based upload feature allows easy import of data from other applications and data stores. CHAKRA can also consume data from other applications that are exposed through XML/SOAP Webservices.

5 Security Architecture

Some of the highlights of CHAKRA's security architecture are:

Authentication/Identification: Uses Container Managed Authentication scheme supported by major Java EE Application Servers. Supported user repositories include LDAP, Native Operating System Users and User Data stored at a relational database.

Authorization: Supports fine grained access control based on roles configured in CHAKRA database. With role based access, it is possible to control visibility of data on a per user basis.

Privacy & Encryption: It is assumed that CHAKRA will run inside corporate firewall. Thus, all data within system is considered protected. At presentation layer, encryption is supported by using HTTPS protocol