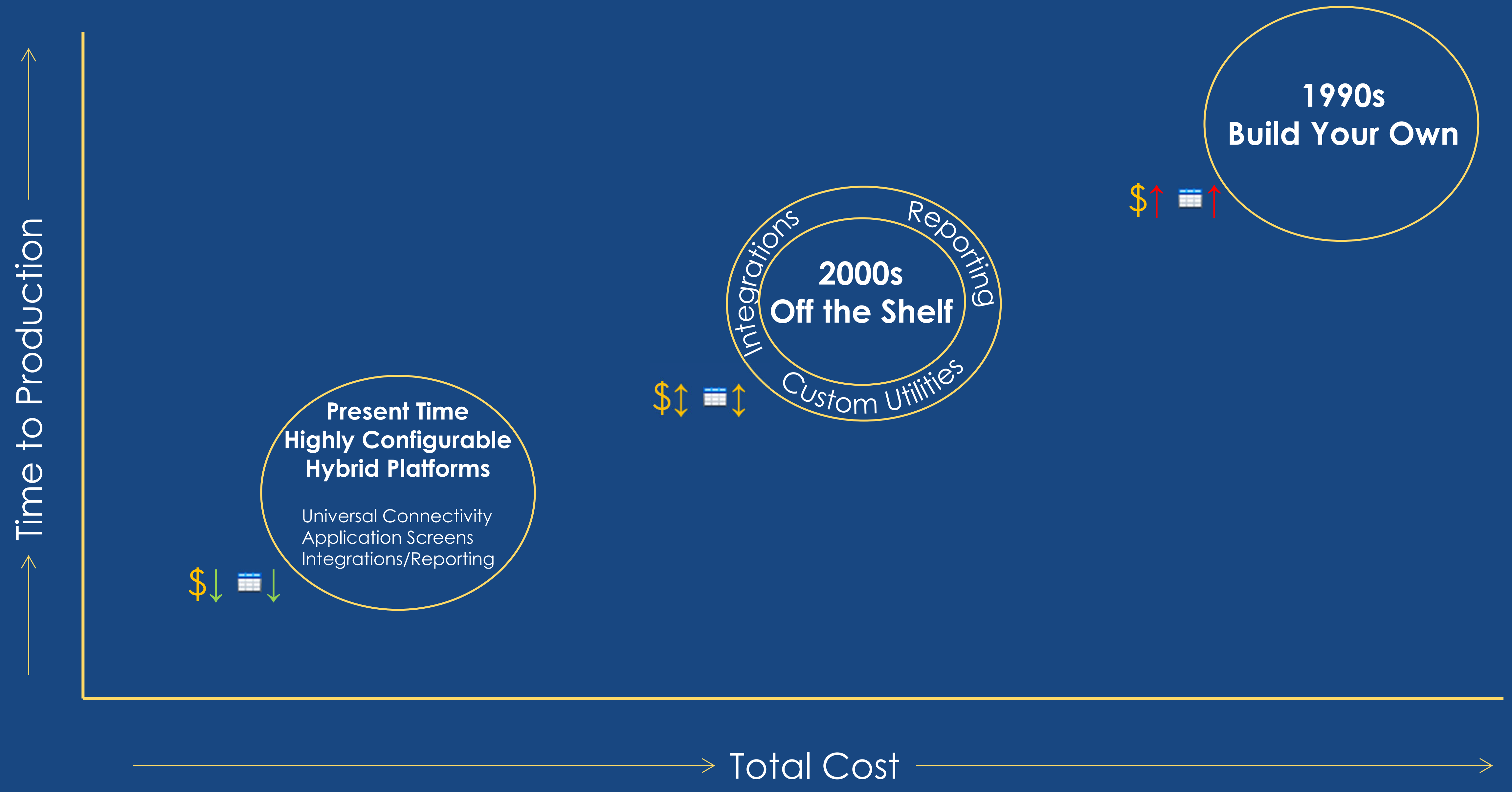
JAR Platform

JORVAK

Technology Solution Evolution



Time to Production

Total Cost

1990s
Build Your Own

$↑ ▦↑

2000s
Off the Shelf

Integrations
Reporting
Custom Utilities

$↕ ▦↕

Present Time
Highly Configurable
Hybrid Platforms

Universal Connectivity
Application Screens
Integrations/Reporting

$↓ ▦↓

What is JAR Platform?

- Jorvak Analytics and Reporting (JAR) is a web based cloud ready integrated toolkit for:
  - **Data Integration** – Database, AWS, SSH, FTP, Network Shares, Loaders, Extractors and more
  - **Application Screens** - Front end UI creation with full role based row/column level security
  - **Analytics and reporting** – HTML Reporting with drillable visualizations and Full Excel based templates for Data presentation/visualization with pivot tables/charts, slicers etc.
  - **Cloud Management** – Integration with Microsoft Azure (AD, SQL etc.), Amazon Web Services (S3, Redshift etc.) and Google Analytics for tracking and analyzing web traffic

- JAR can connect to any database which has JDBC driver available (Nearly all databases have a Java driver) Currently tested on popular databases like Hadoop Hive, Cassandra, SAP HANA, Oracle, Exadata, Teradata, SQL Server, AWS Redshift, IBM DB2, MySQL, PostgreSQL and more

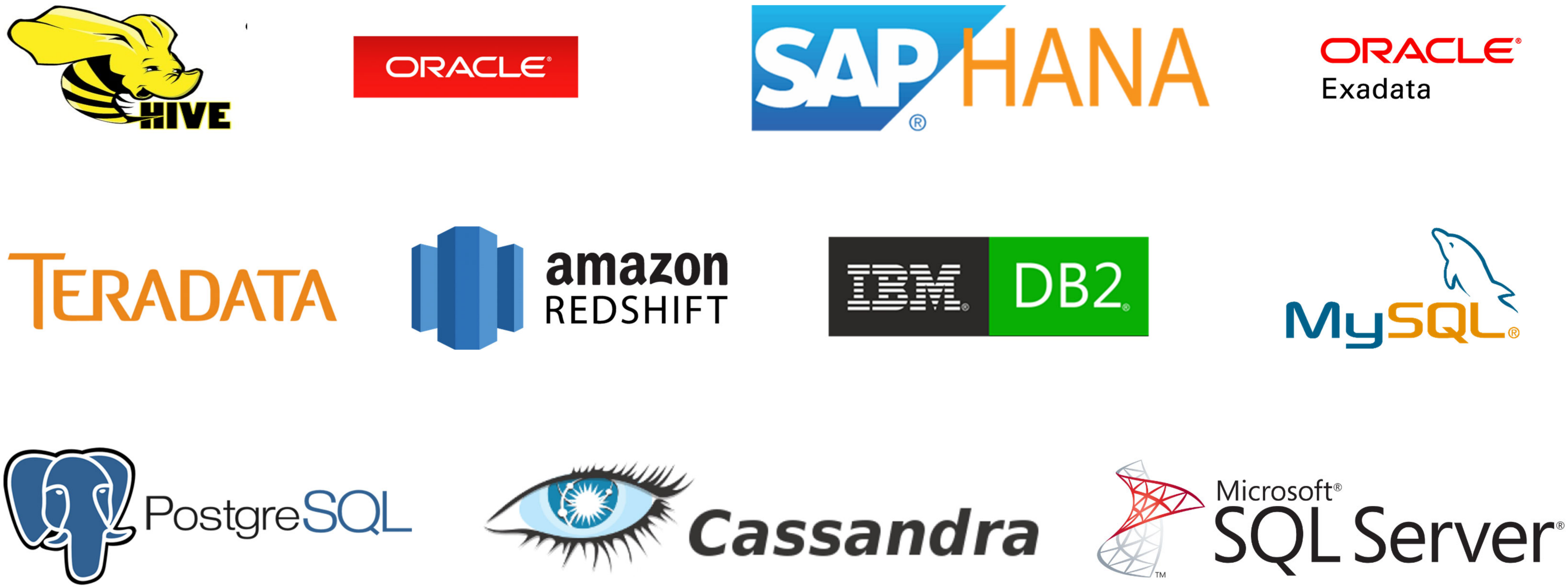
- JAR utilizes open source/common technologies (Excel, SQL, XML, HTML, CSS and JavaScript)

- JAR configuration is set up through XML or Properties files, no Java coding is required

- JAR capabilities can seamlessly be combined to enable complete solutions

Universal Connectivity

JDBC
Java Database Connectivity



Tested connectivity with the above, but JAR connects to anything with a JDBC driver (over 99%+ of all databases)

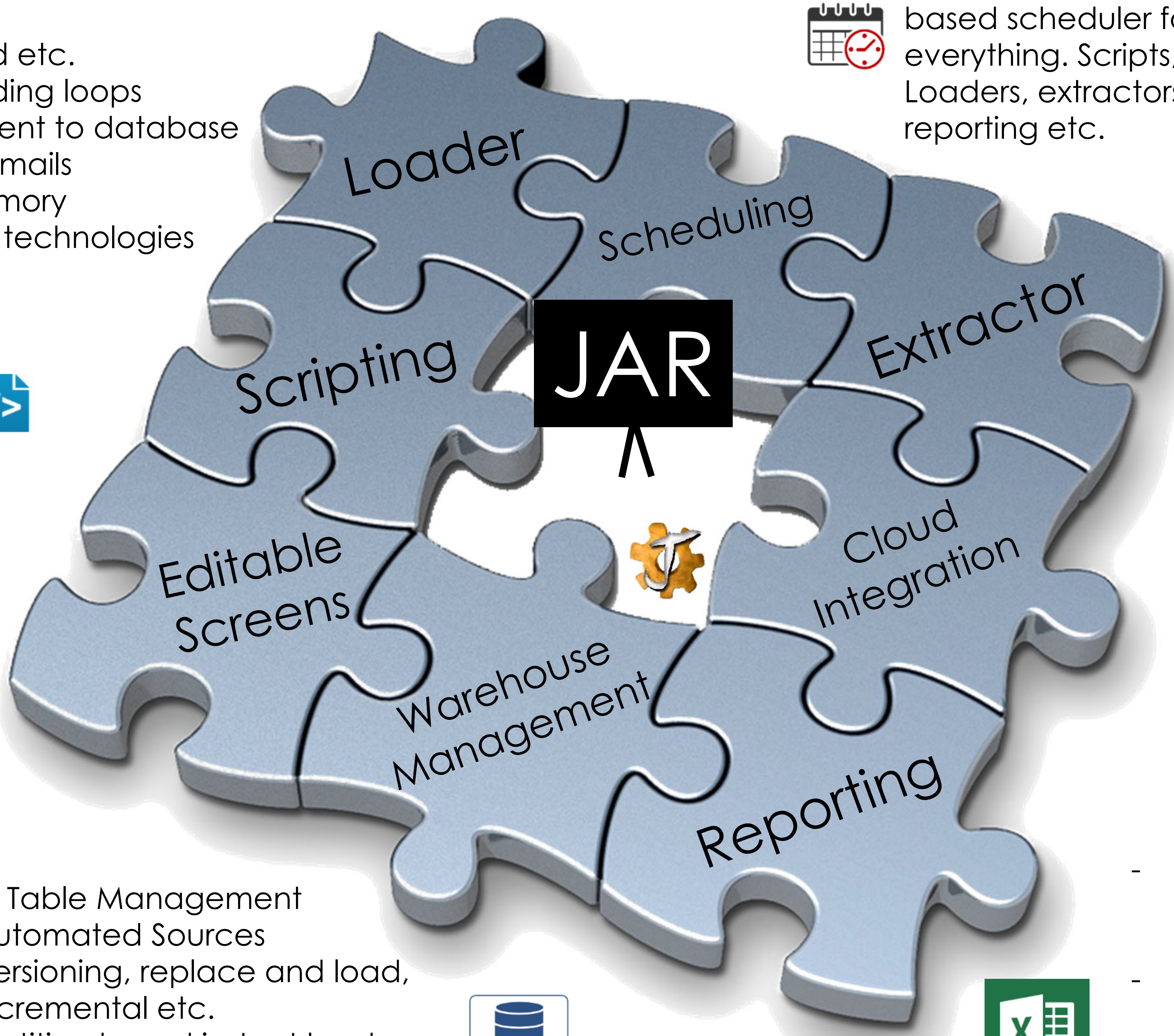
JAR Integrated Architecture

Load any format

- Flat Files – CSVs, Fixed etc.
- Full EDI support including loops
- Stream XLSX file content to database
- Outlook Calendars/Emails
- DB to DB using in memory conversion between technologies

Built in dependency based scheduler for everything. Scripts, Loaders, extractors, reporting etc.

Export Data in any format: CSV, Tab, Custom etc.
- Adhoc or scheduled



- Integrations Simplified
- Full replacement for shell scripting with API, AWS, Checksum, Database, FTP, LDAP, Network Shares and SSH commands.
- Automated error management and notifications

Full integration with Microsoft Azure (AD, SQL etc.), Google Analytics and Amazon Web Services S3 storage and Redshift Data Warehouses

Fully configurable data management front end UI screens built entirely using XML configuration with full row/col level role based security

Fact Table Management

- Automated Sources
- Versioning, replace and load, incremental etc.
- Partition based instant loads
- Partition based built in archiving

- HTML Reporting with built in left and right click drillable visualizations (graphs etc.)
- Full Excel template based streaming exports and dashboards
- EDI file generation via export

JAR Features and Capabilities

Enterprise Features

- ✓ High Performance
- ✓ Security
- ✓ Scheduling of Jobs / Processes
- ✓ Scalable
- ✓ SaaS Auto Update Delivery
- ✓ Data Warehouse
- ✓ Native Connections
- ✓ XML Based Configuration
- ✓ Time to Deliver
- ✓ Cost

Reduce Development Time

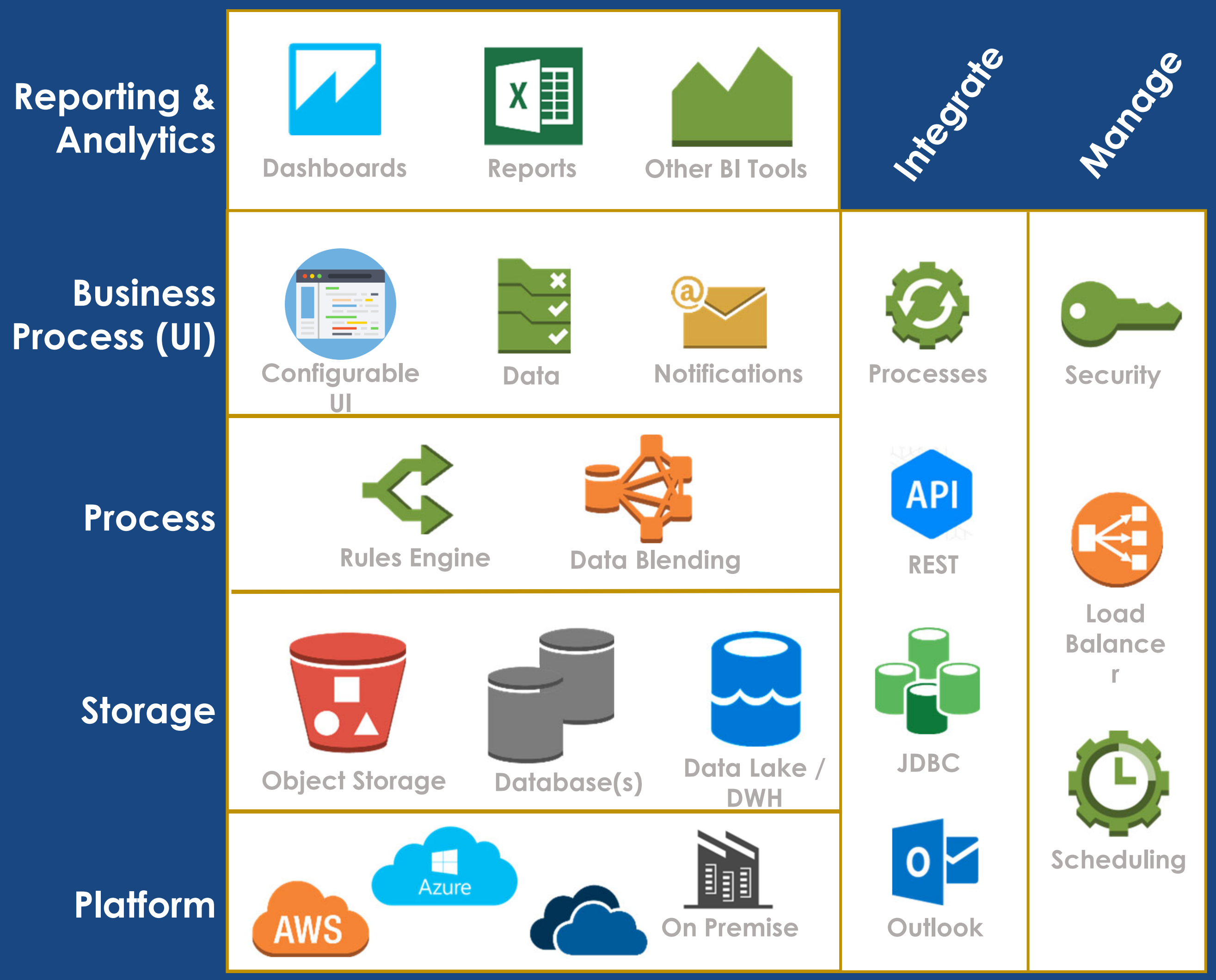
- Cut integration development and time by ~33%
- Leverage client analysts and eliminated the need for developers
- Reduced SDLC Documentation

Performance (Examples)

- Excel Export Performance without significant memory usage:
  - ✓ Extract 100,000 rows x 50 cols wide (5 Million cells) in 45 secs
  - ✓ Extract 1 Million rows x 50 cols wide (50 Million cells) in 6 mins
  - ✓ Extract 802K rows x 149 cols (119.5 Million cells) in 11.5 minutes
- Load into Oracle: 237M rows x 165 cols (39.1 Billion cells) in 1 hr. 14 secs.
- Extracted from Database, zipped and sent to AWS S3
  - ✓ Teradata – 7.3 Million rows x 15 cols wide (109 Million cells) in 7 mins
  - ✓ Oracle - 276M rows x 151 cols (41.6 Billion cells) in 4 hrs. 51 mins.

Rapid Reporting

- Rapidly create reports using Excel templates for executive dashboards, reporting and application screens supporting graphs, pivot tables and pivot charts and macros embedded directly in the Excel template.



Integrate

Manage

Reporting & Analytics: Dashboards, Reports, Other BI Tools

Business Process (UI): Configurable UI, Data, Notifications

Process: Rules Engine, Data Blending

Storage: Object Storage, Database(s), Data Lake / DWH

Platform: AWS, Azure, On Premise

Integrate: Processes, REST, JDBC, Outlook

Manage: Security, Load Balancer, Scheduling

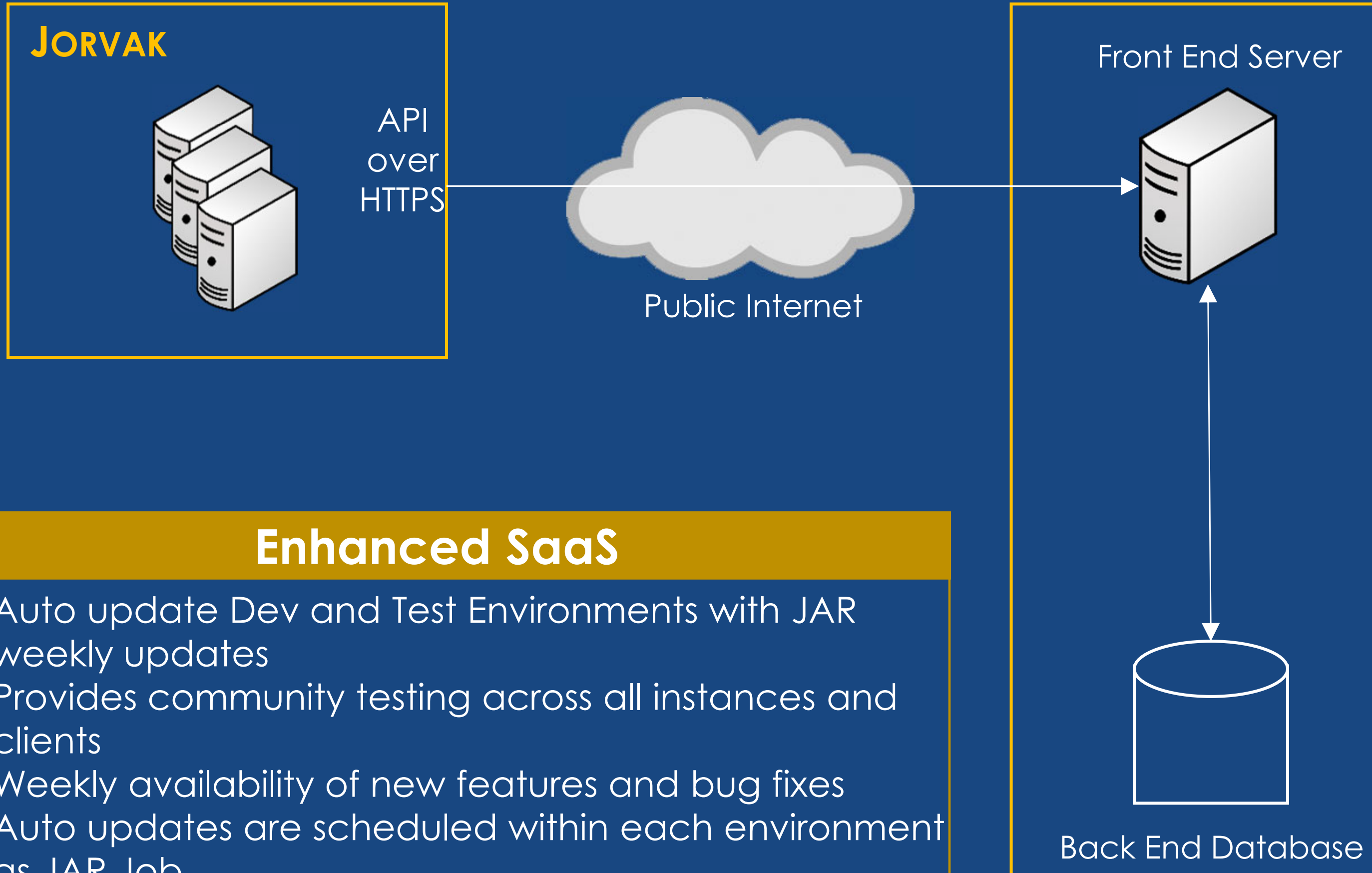
JAR Platform Wide Automation

Auto Update Platform – Scheduled fully automated self update capability

Self Service Workflow based Security – Account and Requests are full automated self service with assigned approvers for each security role

Integrated Help Desk/Ticketing – 2 click screenshot based help requests. Fully integrated IT ticket management for two way requestor and responder communication

JAR SaaS Fully Automated Update Delivery using RPA



JAR RPA - Robotic Process Automation (Scheduled Job)

1. Connect to Jorvak API over HTTPS and download encrypted release zip file
2. Decrypt (using password) and extract release zip file
3. Load changed and new path/filenames/checksum to release stage table
4. Backup current changed and new files from existing environment
5. Compute checksum for all files in this environment
6. Compare environment checksum with downloaded checksum to verify files weren't overwritten in this environment
   a. If checksums match, then replace existing files with newly downloaded files
   b. If checksums don't match, then replace existing files and mark for CoE/client IT manual review
7. Run the release JAR script to update properties and database
8. Redeploy the Tomcat (or other Java App Server) application
9. Send release update summary email to CoE/client IT and Jorvak support

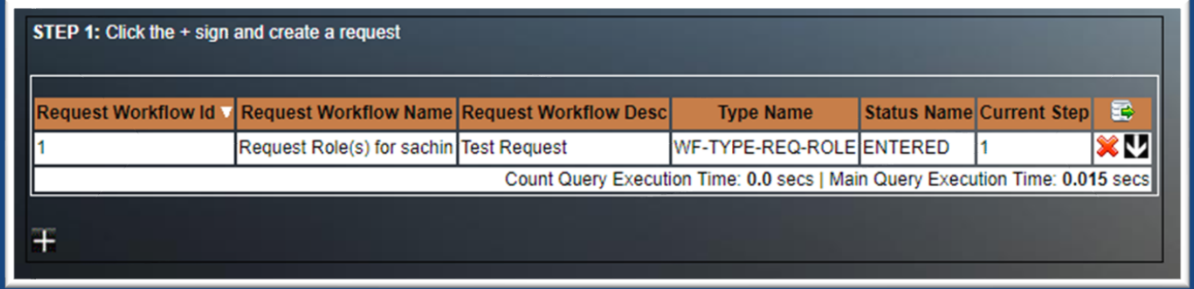
Client Environment

Enhanced SaaS

- Auto update Dev and Test Environments with JAR weekly updates
- Provides community testing across all instances and clients
- Weekly availability of new features and bug fixes
- Auto updates are scheduled within each environment as JAR Job

JAR Security Automation and 2-Click Help Request

Workflow based Security Account/Role Management

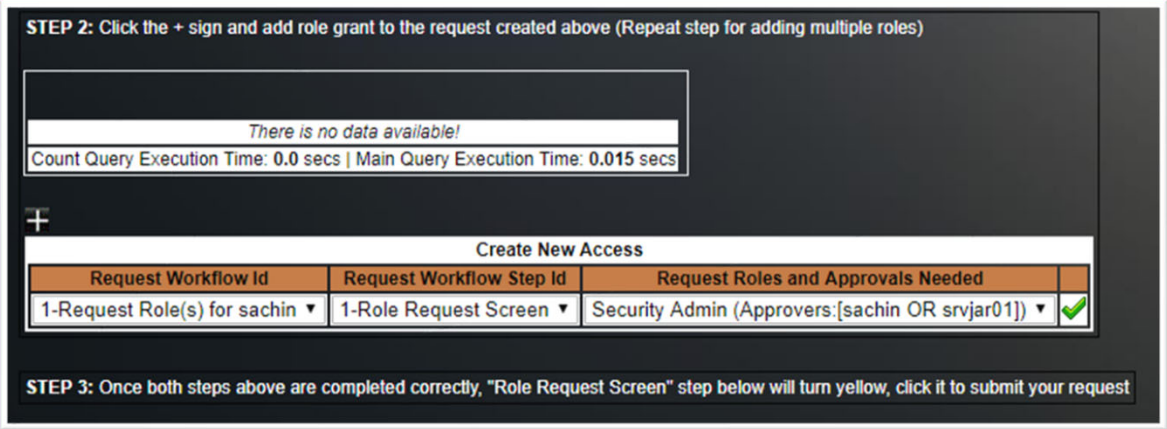


STEP 1: Click the + sign and create a request

Request Workflow Id	Request Workflow Name	Request Workflow Desc	Type Name	Status Name	Current Step	
1	Request Role(s) for sachin	Test Request	WF-TYPE-REQ-ROLE	ENTERED	1	✗ ↓

Count Query Execution Time: 0.0 secs | Main Query Execution Time: 0.015 secs

+



STEP 2: Click the + sign and add role grant to the request created above (Repeat step for adding multiple roles)

There is no data available!

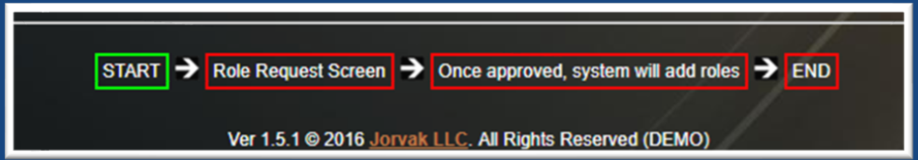
Count Query Execution Time: 0.0 secs | Main Query Execution Time: 0.015 secs

+

Create New Access

Request Workflow Id	Request Workflow Step Id	Request Roles and Approvals Needed	
1-Request Role(s) for sachin ▾	1-Role Request Screen ▾	Security Admin (Approvers:[sachin OR srvjar01]) ▾	✓

STEP 3: Once both steps above are completed correctly, "Role Request Screen" step below will turn yellow, click it to submit your request

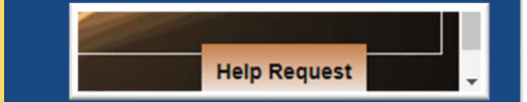


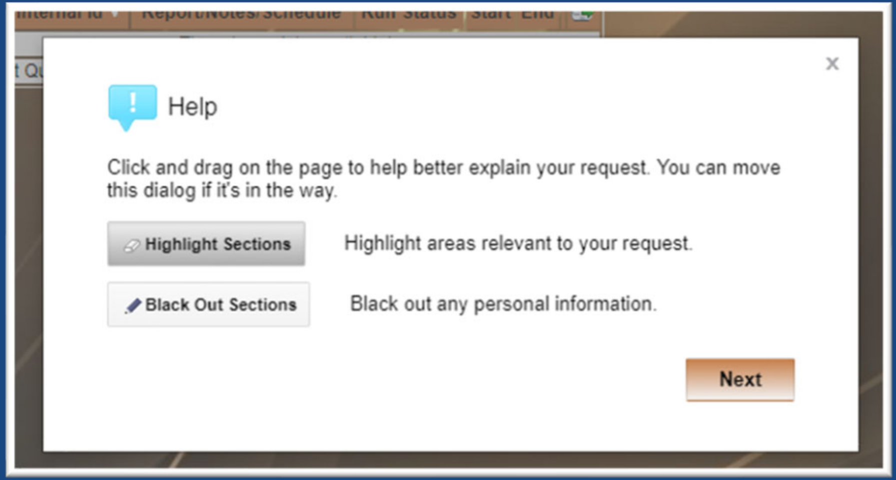
START → Role Request Screen → Once approved, system will add roles → END

Ver 1.5.1 © 2016 Jorvak LLC. All Rights Reserved (DEMO)

No IT or operational involvement for security account/role management. Significantly simplifies audit and DevOps

Screenshot based 2-click built in Help Request



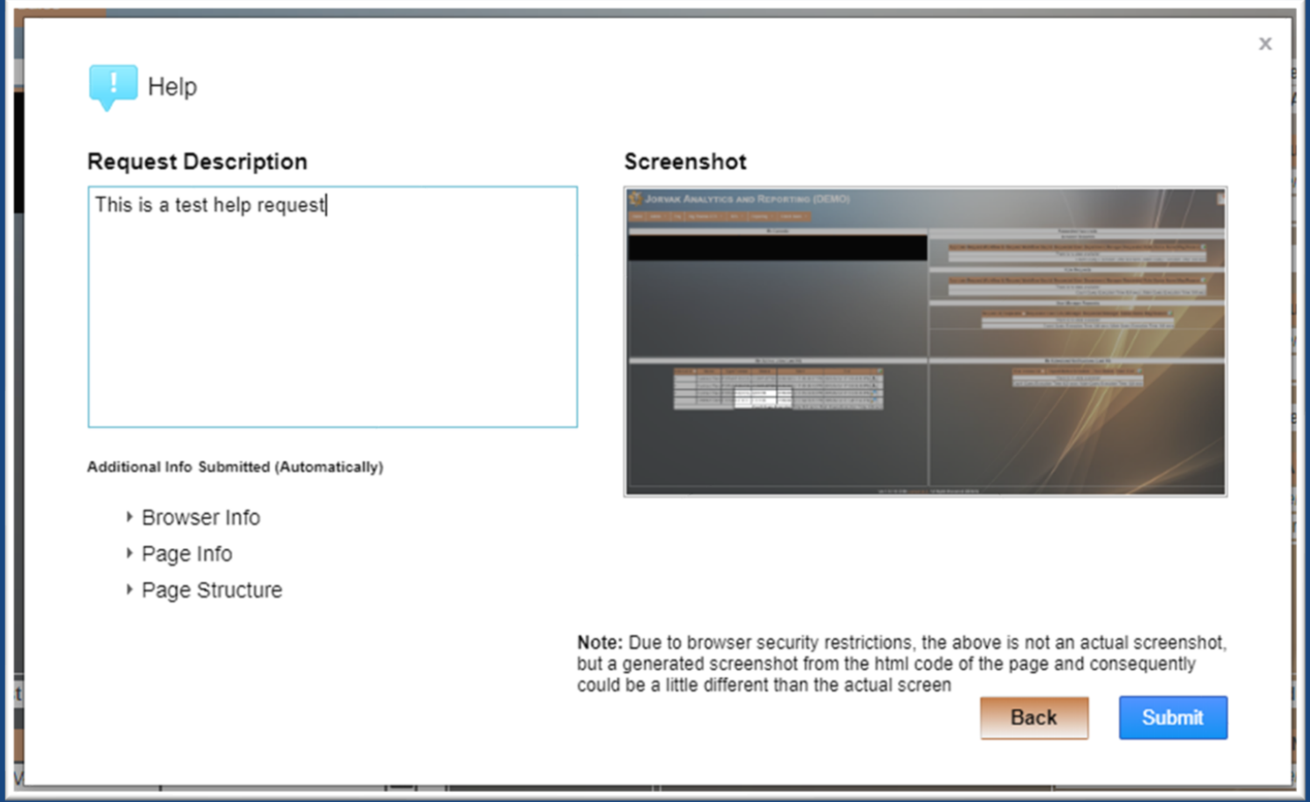


Help

Click and drag on the page to help better explain your request. You can move this dialog if it's in the way.

Highlight Sections — Highlight areas relevant to your request.

Black Out Sections — Black out any personal information.



Help

Request Description

This is a test help request

Screenshot

Additional Info Submitted (Automatically)

- Browser Info
- Page Info
- Page Structure

Note: Due to browser security restrictions, the above is not an actual screenshot, but a generated screenshot from the html code of the page and consequently could be a little different than the actual screen

JAR Features and Capabilities

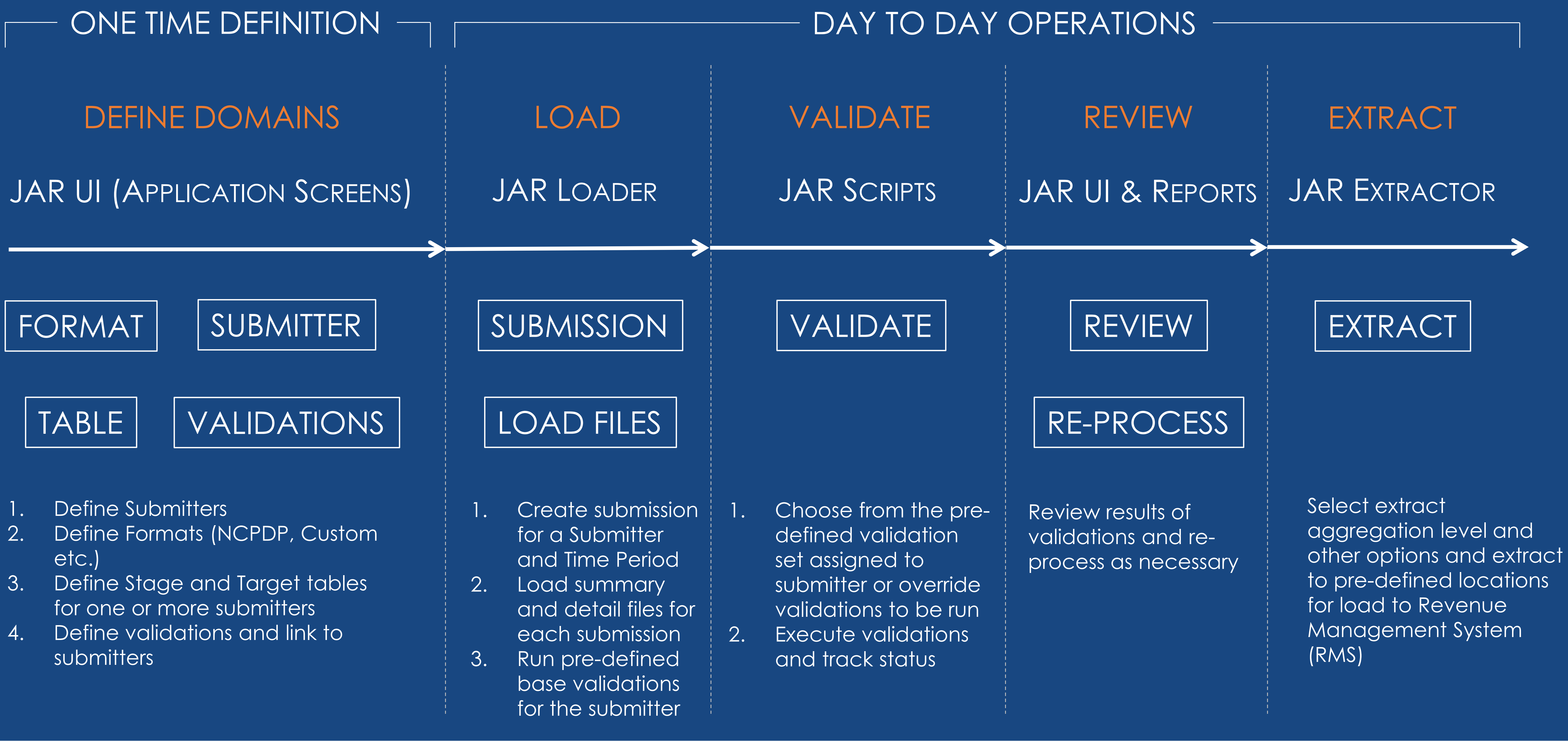
Component	Example Alternatives
Reporting & Analytics	Cognos, Qlik, Tableau, Power BI
Business Process (UI)	UI Frameworks & BPM Solutions (Tibco, Appian, etc.)
Integration	Talend, Informatica, Pentaho Data Integration
Scheduling	Autosys Scheduler, Quartz Scheduler, Spring Batch
Scripting	Ansible, Chef, Puppet, PowerShell
Process/Extractor	Talend, Informatica, Pentaho Data Integration
Warehouse Mgmt	Native to Data Warehouse features and many third party add-ons

JAR Implementations at TOP 5 Pharma Company

Application Component	Data Steward	Membership Analytics	Cloud Analytics	Medicaid Liability	Analytic Reporting	Payments	Unstructured Data Analytics	Cloud Integrations	Robotic Process Automation
Editing	✓	✓	✓	✓		✓	✓		
Loading		✓	✓	✓		✓	✓		✓
Extracting			✓	✓		✓	✓	✓	✓
Scheduling	✓		✓			✓		✓	✓
Reporting	✓	✓	✓	✓	✓	✓	✓	✓	✓
Scripting			✓	✓		✓			✓
Cloud Integration			✓					✓	
Warehouse Management			✓					✓	
Help Request Management	✓	✓	✓	✓	✓	✓	✓		

Sample Application – Submission Validation System (SVS)

- HORIZONTALLY SCALABLE – PARALLEL VALIDATION EXECUTION
- BEST OF BREED DATABASE TECHNOLOGIES



ONE TIME DEFINITION

DEFINE DOMAINS

JAR UI (Application Screens)

FORMAT

SUBMITTER

TABLE

VALIDATIONS

1. Define Submitters
2. Define Formats (NCPDP, Custom etc.)
3. Define Stage and Target tables for one or more submitters
4. Define validations and link to submitters

DAY TO DAY OPERATIONS

LOAD

JAR Loader

SUBMISSION

LOAD FILES

1. Create submission for a Submitter and Time Period
2. Load summary and detail files for each submission
3. Run pre-defined base validations for the submitter

VALIDATE

JAR Scripts

VALIDATE

1. Choose from the pre-defined validation set assigned to submitter or override validations to be run
2. Execute validations and track status

REVIEW

JAR UI & Reports

REVIEW

RE-PROCESS

Review results of validations and re-process as necessary

EXTRACT

JAR Extractor

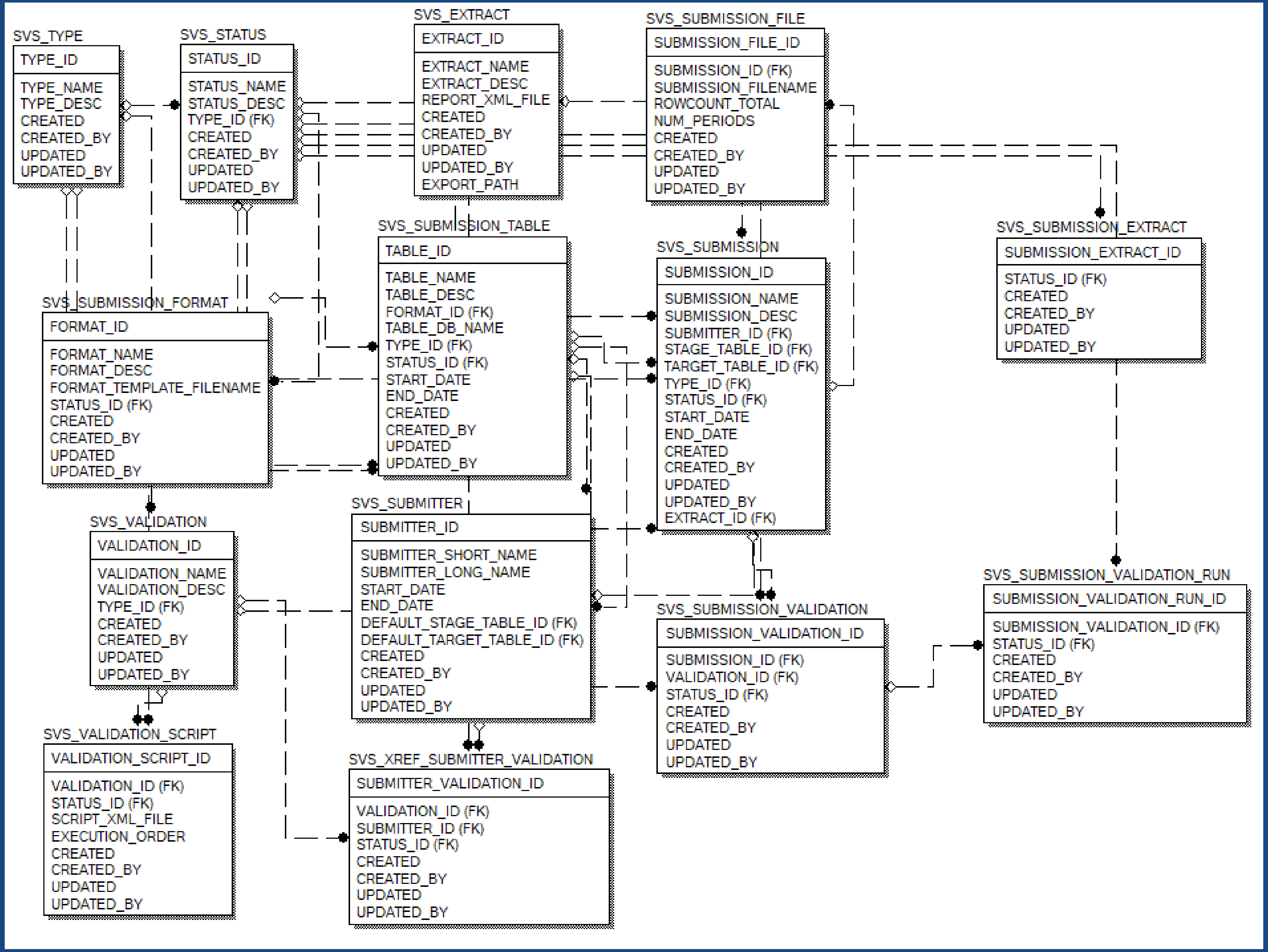
EXTRACT

Select extract aggregation level and other options and extract to pre-defined locations for load to Revenue Management System (RMS)

Submission Validation System (SVS) – ER Diagram

Sample Database schema design

- Define
- Load
- Validate
- Review
- Extract



SVS_TYPE
TYPE_ID
TYPE_NAME
TYPE_DESC
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_STATUS
STATUS_ID
STATUS_NAME
STATUS_DESC
TYPE_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_EXTRACT
EXTRACT_ID
EXTRACT_NAME
EXTRACT_DESC
REPORT_XML_FILE
CREATED
CREATED_BY
UPDATED
UPDATED_BY
EXPORT_PATH

SVS_SUBMISSION_FILE
SUBMISSION_FILE_ID
SUBMISSION_ID (FK)
SUBMISSION_FILENAME
ROWCOUNT_TOTAL
NUM_PERIODS
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_SUBMISSION_TABLE
TABLE_ID
TABLE_NAME
TABLE_DESC
FORMAT_ID (FK)
TABLE_DB_NAME
TYPE_ID (FK)
STATUS_ID (FK)
START_DATE
END_DATE
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_SUBMISSION
SUBMISSION_ID
SUBMISSION_NAME
SUBMISSION_DESC
SUBMITTER_ID (FK)
STAGE_TABLE_ID (FK)
TARGET_TABLE_ID (FK)
TYPE_ID (FK)
STATUS_ID (FK)
START_DATE
END_DATE
CREATED
CREATED_BY
UPDATED
UPDATED_BY
EXTRACT_ID (FK)

SVS_SUBMISSION_EXTRACT
SUBMISSION_EXTRACT_ID
STATUS_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_SUBMISSION_FORMAT
FORMAT_ID
FORMAT_NAME
FORMAT_DESC
FORMAT_TEMPLATE_FILENAME
STATUS_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_VALIDATION
VALIDATION_ID
VALIDATION_NAME
VALIDATION_DESC
TYPE_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_SUBMITTER
SUBMITTER_ID
SUBMITTER_SHORT_NAME
SUBMITTER_LONG_NAME
START_DATE
END_DATE
DEFAULT_STAGE_TABLE_ID (FK)
DEFAULT_TARGET_TABLE_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_SUBMISSION_VALIDATION_RUN
SUBMISSION_VALIDATION_RUN_ID
SUBMISSION_VALIDATION_ID (FK)
STATUS_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

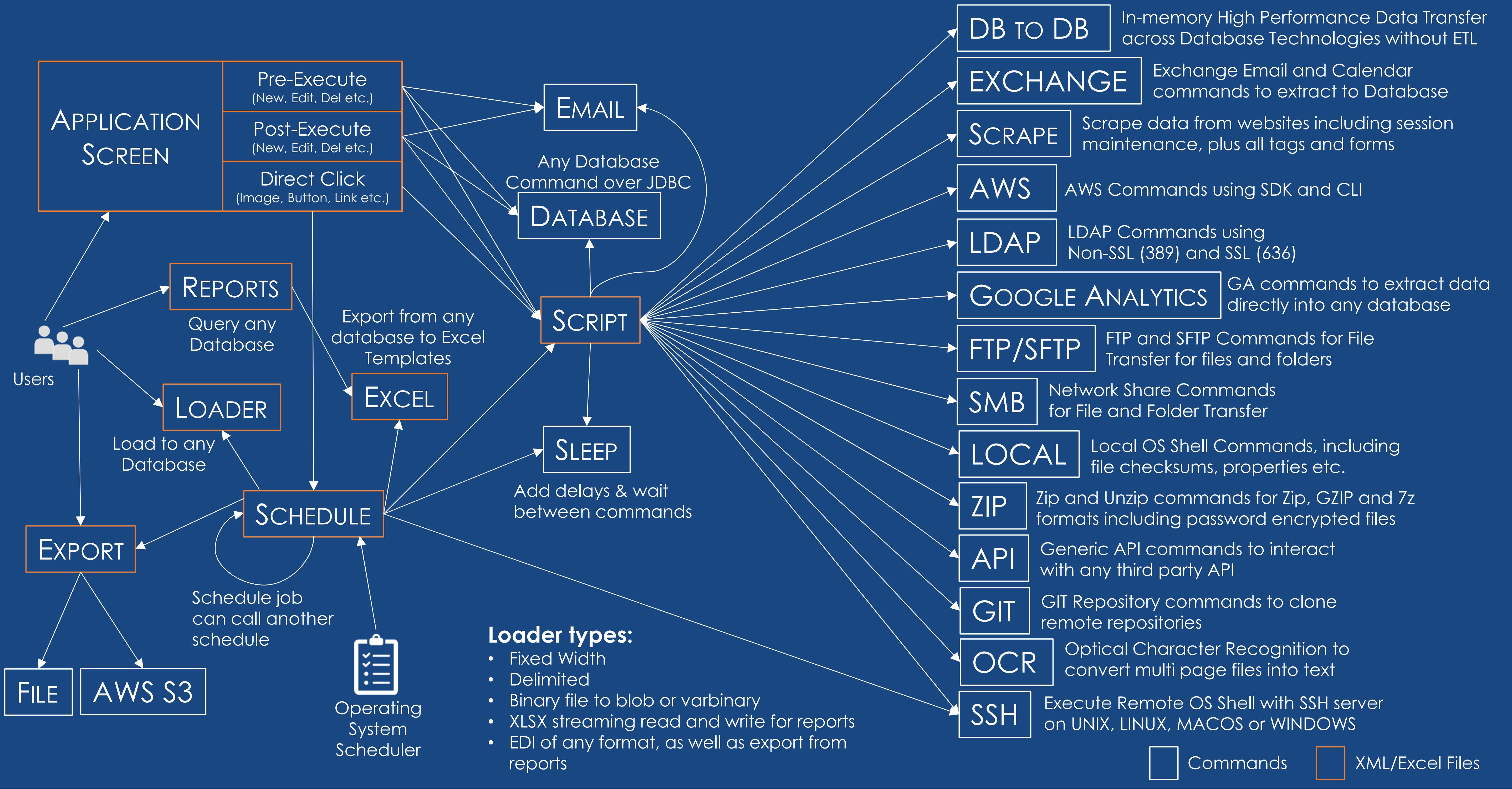
SVS_SUBMISSION_VALIDATION
SUBMISSION_VALIDATION_ID
SUBMISSION_ID (FK)
VALIDATION_ID (FK)
STATUS_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_VALIDATION_SCRIPT
VALIDATION_SCRIPT_ID
VALIDATION_ID (FK)
STATUS_ID (FK)
SCRIPT_XML_FILE
EXECUTION_ORDER
CREATED
CREATED_BY
UPDATED
UPDATED_BY

SVS_XREF_SUBMITTER_VALIDATION
SUBMITTER_VALIDATION_ID
VALIDATION_ID (FK)
SUBMITTER_ID (FK)
STATUS_ID (FK)
CREATED
CREATED_BY
UPDATED
UPDATED_BY

Anatomy of a JAR Application

All Configuration Done via HTML, JS, CSS, XML and SQL



Application Screen

Pre-Execute (New, Edit, Del etc.)

Post-Execute (New, Edit, Del etc.)

Direct Click (Image, Button, Link etc.)

Email

Any Database Command over JDBC

Database

Script

Sleep

Add delays & wait between commands

Users

Reports

Query any Database

Export from any database to Excel Templates

Excel

Loader

Load to any Database

Schedule

Schedule job can call another schedule

Export

File

AWS S3

Operating System Scheduler

Loader types:

- Fixed Width
- Delimited
- Binary file to blob or varbinary
- XLSX streaming read and write for reports
- EDI of any format, as well as export from reports

DB to DB

In-memory High Performance Data Transfer across Database Technologies without ETL

Exchange

Exchange Email and Calendar commands to extract to Database

Scrape

Scrape data from websites including session maintenance, plus all tags and forms

AWS

AWS Commands using SDK and CLI

LDAP

LDAP Commands using Non-SSL (389) and SSL (636)

Google Analytics

GA commands to extract data directly into any database

FTP/SFTP

FTP and SFTP Commands for File Transfer for files and folders

SMB

Network Share Commands for File and Folder Transfer

LOCAL

Local OS Shell Commands, including file checksums, properties etc.

ZIP

Zip and Unzip commands for Zip, GZIP and 7z formats including password encrypted files

API

Generic API commands to interact with any third party API

GIT

GIT Repository commands to clone remote repositories

OCR

Optical Character Recognition to convert multi page files into text

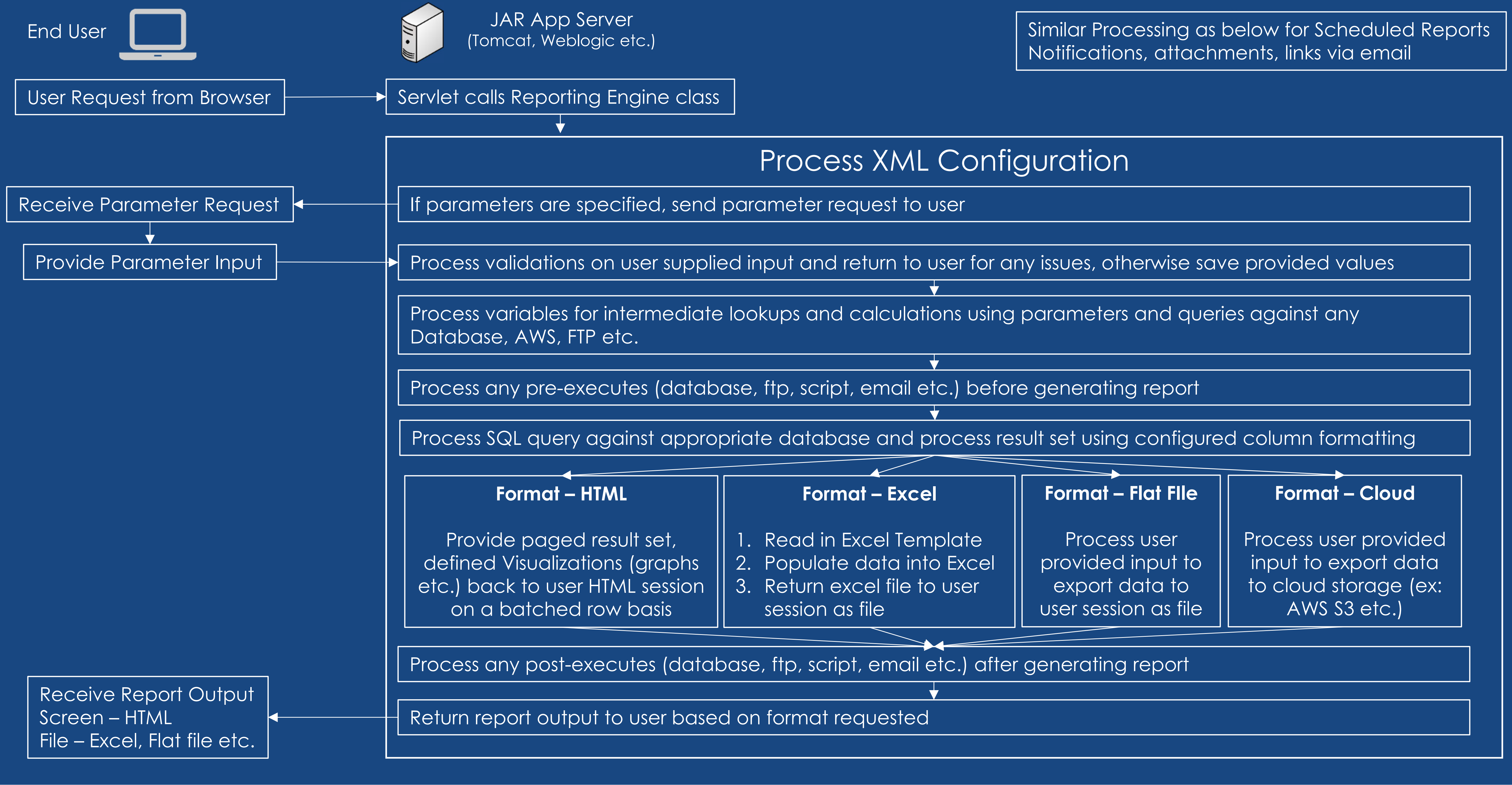
SSH

Execute Remote OS Shell with SSH server on UNIX, LINUX, MACOS or WINDOWS

□ Commands

□ XML/Excel Files

JAR Configurability – Reporting Engine



End User

JAR App Server
(Tomcat, Weblogic etc.)

Similar Processing as below for Scheduled Reports
Notifications, attachments, links via email

User Request from Browser

Servlet calls Reporting Engine class

Process XML Configuration

If parameters are specified, send parameter request to user

Receive Parameter Request

Provide Parameter Input

Process validations on user supplied input and return to user for any issues, otherwise save provided values

Process variables for intermediate lookups and calculations using parameters and queries against any Database, AWS, FTP etc.

Process any pre-executes (database, ftp, script, email etc.) before generating report

Process SQL query against appropriate database and process result set using configured column formatting

Format – HTML

Provide paged result set, defined Visualizations (graphs etc.) back to user HTML session on a batched row basis

Format – Excel

1. Read in Excel Template
2. Populate data into Excel
3. Return excel file to user session as file

Format – Flat FIle

Process user provided input to export data to user session as file

Format – Cloud

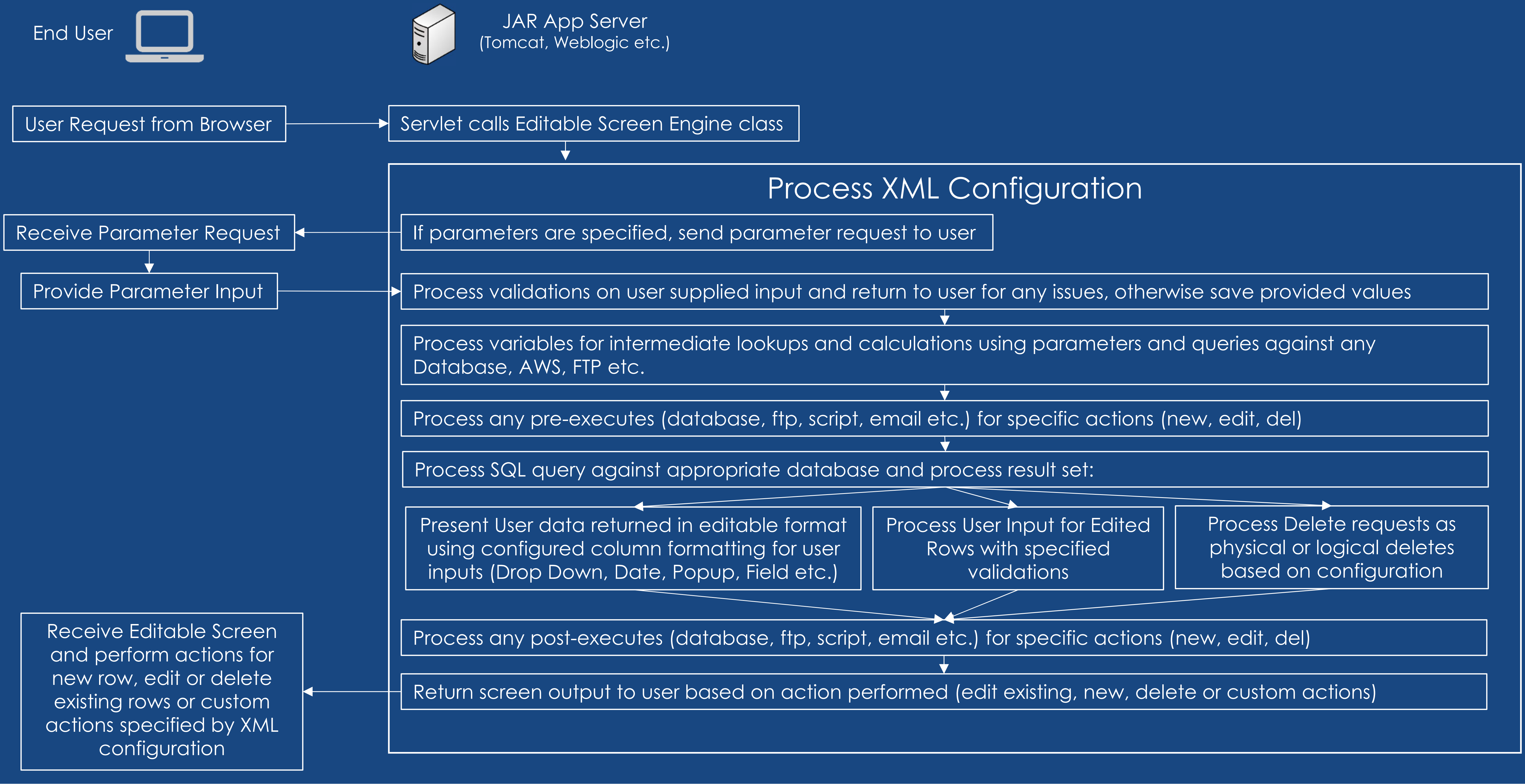
Process user provided input to export data to cloud storage (ex: AWS S3 etc.)

Process any post-executes (database, ftp, script, email etc.) after generating report

Return report output to user based on format requested

Receive Report Output
Screen – HTML
File – Excel, Flat file etc.

JAR Configurability – Editable Screen Engine



End User

JAR App Server
(Tomcat, Weblogic etc.)

User Request from Browser

Servlet calls Editable Screen Engine class

Process XML Configuration

If parameters are specified, send parameter request to user

Receive Parameter Request

Provide Parameter Input

Process validations on user supplied input and return to user for any issues, otherwise save provided values

Process variables for intermediate lookups and calculations using parameters and queries against any Database, AWS, FTP etc.

Process any pre-executes (database, ftp, script, email etc.) for specific actions (new, edit, del)

Process SQL query against appropriate database and process result set:

Present User data returned in editable format using configured column formatting for user inputs (Drop Down, Date, Popup, Field etc.)

Process User Input for Edited Rows with specified validations

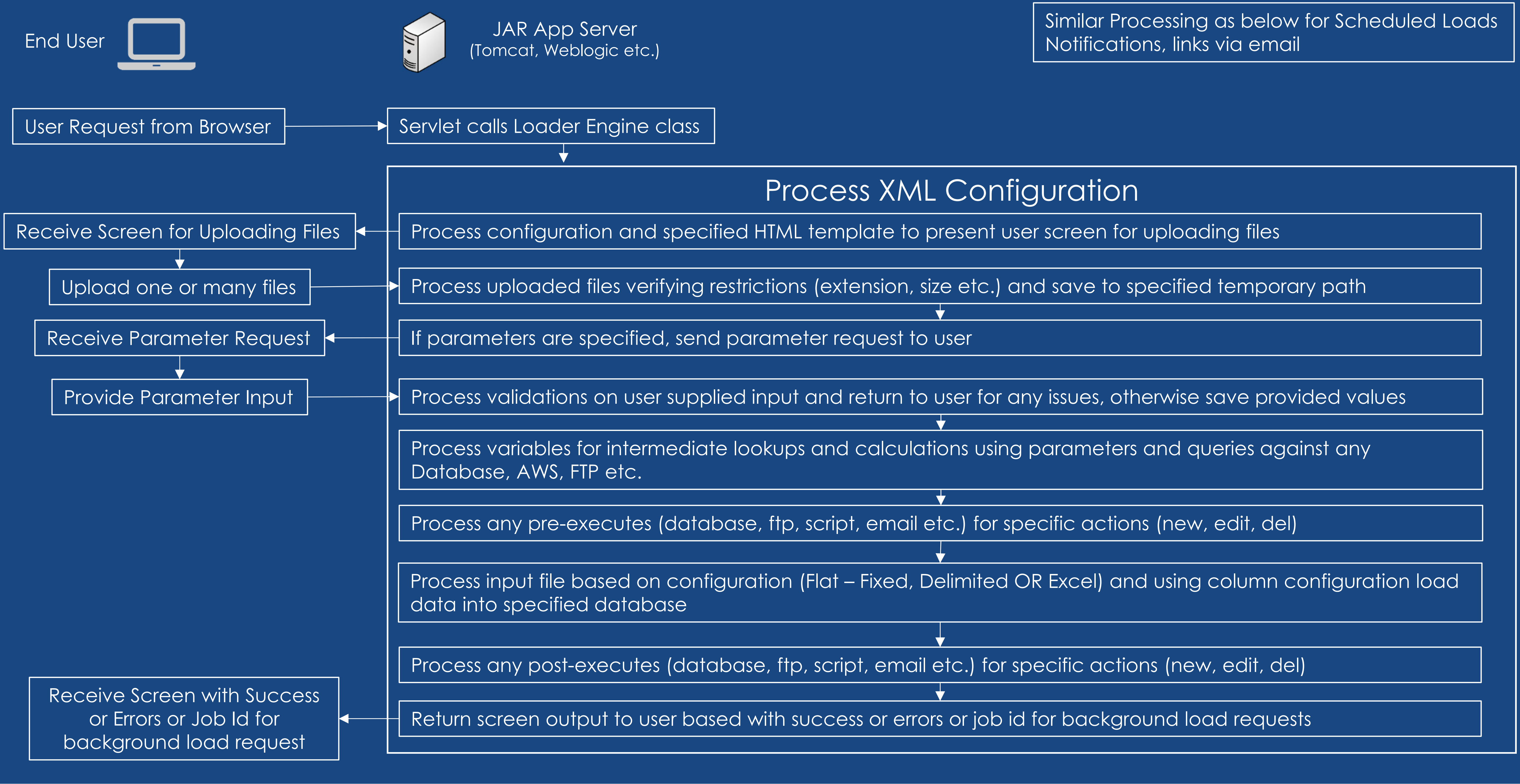
Process Delete requests as physical or logical deletes based on configuration

Process any post-executes (database, ftp, script, email etc.) for specific actions (new, edit, del)

Return screen output to user based on action performed (edit existing, new, delete or custom actions)

Receive Editable Screen and perform actions for new row, edit or delete existing rows or custom actions specified by XML configuration

JAR Configurability – Loader Engine



End User

JAR App Server
(Tomcat, Weblogic etc.)

Similar Processing as below for Scheduled Loads
Notifications, links via email

User Request from Browser

Servlet calls Loader Engine class

Process XML Configuration

Receive Screen for Uploading Files

Process configuration and specified HTML template to present user screen for uploading files

Upload one or many files

Process uploaded files verifying restrictions (extension, size etc.) and save to specified temporary path

Receive Parameter Request

If parameters are specified, send parameter request to user

Provide Parameter Input

Process validations on user supplied input and return to user for any issues, otherwise save provided values

Process variables for intermediate lookups and calculations using parameters and queries against any Database, AWS, FTP etc.

Process any pre-executes (database, ftp, script, email etc.) for specific actions (new, edit, del)

Process input file based on configuration (Flat – Fixed, Delimited OR Excel) and using column configuration load data into specified database

Process any post-executes (database, ftp, script, email etc.) for specific actions (new, edit, del)

Receive Screen with Success
or Errors or Job Id for
background load request

Return screen output to user based with success or errors or job id for background load requests

JAR Simple Technology Stack

Only 3 items required: Tomcat, JVM and a Database Schema

Front End Server

Required

- **Tomcat** App Server
- **JVM** – Java 1.8 or Open JDK

Optional

- AWS CLI
- Azure CLI
- Oracle Client
- SQL Server BCP CLI

Back End Database

Schema/Database

- Oracle

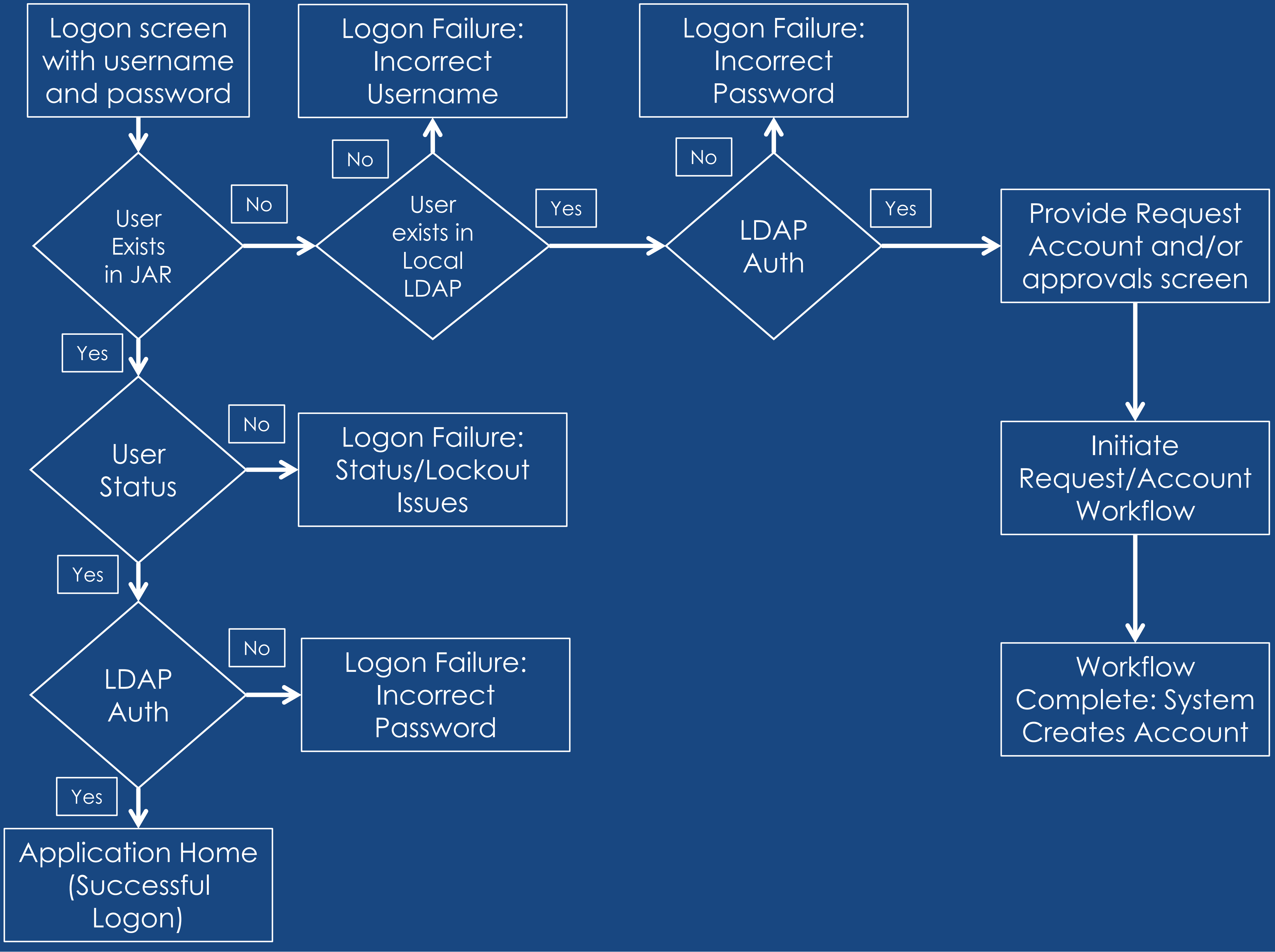
OR

- SQL Server

Note: Can be a managed cloud database service such as AWS RDS or Azure SQL Service

JAR Security – Automation – Self Request

Scheduled High Frequency (Hourly, 30 mins, 15 mins etc.) JAR Script Job to maintain local copy of LDAP



Logon screen with username and password

User Exists in JAR

No

User exists in Local LDAP

No

Logon Failure: Incorrect Username

Yes

LDAP Auth

No

Logon Failure: Incorrect Password

Yes

Provide Request Account and/or approvals screen

Initiate Request/Account Workflow

Workflow Complete: System Creates Account

Yes

User Status

No

Logon Failure: Status/Lockout Issues

Yes

LDAP Auth

No

Logon Failure: Incorrect Password

Yes

Application Home (Successful Logon)