Outline

Tuesday, October 7, 2014 9:20 AM

<Name> Documentation

What is <Name>?

Overview of Application:
- Purpose, high level summary of the main problem we solve (for perspective of: open source developers). Developed for open source use… Scala and Json…fully customizable, etc.

Capabilities
Currently in Use Examples

Getting Started

New User Guide
Quick Installation Guide (or link to full Installation Guide in API section)
Open Source License (?)
Tutorials

Developer Reference

Overview and Concepts
- Summary
  - <Name> achieves near real-time processing of transactional data stored in the Hadoop Distributed File System. It does this by: <tbd>

Architecture
- Overview (and top level Diagram)
- Architecture
  - Inputs and Outputs: Messages and Alerts
    - Diagrams
  - Description of Engine Processing Flow
    - Diagram
  - Rules as Models
    - Diagrams
  - Why we get the speed we do
- Benefits and Highlights
  - Key Technologies
  - Cluster/Node Solutions
  - Performance and Resource Utilization
  - Scalability, Availability, Optimization

API Documentation
- Installation Guide
  - Download the <Name> installation file here
  - Installation steps
  - *Access steps - Keys and accounts*
- Data Structures
- Function Signatures
  - Names
  - Function parameters names and types
    - Return types
  - Error handling
  - Pre/post conditions
  - Description of state changes post-execution

tbp: compilers…

Objects

Relationship of type to other types (inheritance: super-types, sub-types, implemented interfaces or traits), composite structures, delegating entities or any mixed-in set of functionality

Public parts of an instance of a class:

Constants

Name and type of properties

Method signatures

Class-specific operators? (is overloading supported)

Static nature?

Constraints on objects derived from this class

Nested structures

Object exception handling

Concurrency Handling

Metadata API (if different from the Objects section above)

Object model diagram

Elements

Relationship to other Types

Public Parts

Constraints

Nested Structures

Exception Handling

Concurrency Handling

Models

<sic>

Messages

<sic>

Containers

<sic>

Concepts

<sic>

Fields

<sic>

Communication protocols

Unstable parts?

Deprecated parts?

Frameworks?:

GUI

Network

File System

How to Deploy Client Libraries

Public Data Sets by Industry

Finance

Barclays (may generalize / remove customer name)

Healthcare

Edifecs (may generalize / remove customer name)

Diverse

<other data set>

Glossary

Troubleshooting

SDK Tools / Documentation

Tool 1

Install and User guide for Tool 1

Tool 2

Install and User guide for Tool 2

Release Notes

- *2014 1215 Release Notes*
- *20141201 Release Notes*

- *20141115 Release Notes*

Learning

Step by Step Guide - Create your First <Name> Project
Customization Guide

Licensing Options *(this will likely be somewhere else on our github page, but can keep links to it here)*

Free Version
Enterprise Version
Services

Test Setup Install Guide

Monday, October 27, 2014 7:42 AM

Note: This is Williams's Guide to Edifecs, based on Rich's to Barclay.

Note: * indicates info still needed

_____

Prerequisites:

- Approx *MM* GB free
- GitHub Account & client installed.
  - Install git: http://git-scm.com/downloads
- 7-zip intalled (http://www.7-zip.org/) Choose for 32bit or 64bit.
- Java installed
- Access to: https://github.com/ligaDATA/RTD_ReadOnly OR the RTD install package (.tar??)

Steps to setup and deploy OnLEPManager:

1. Download and install Scala 2.10.4: http://www.scala-lang.org/download/2.10.4.html
2. Download and install sbt: http://www.scala-sbt.org/download.html
3. Make sure Scala and sbt paths are in the PATH environmental variable

If you have GitHub access to https://github.com/ligaDATA/RTD_ReadOnly

4. Download source code from https://github.com/ligaDATA/RTD_ReadOnly

Mac & Linux users:

5. Execute either "sudo apt-get install git-core" or "sudo yum install git" depending on your flavor.
6. Navigate to the folder you wish to clone the source to
   a. In command line execute the following: git clone https://github.com/ligaDATA/RTD_ReadOnly
7. Run install script: bash /path/to/source/trunk/SampleApplication/Medical/bin/InstallOnLEP_Medical /path/to/install/location /path/to/source/trunk
   a. This is a bash script usable on linux and mac. If you want to run on windows, it will need minor changes.

Windows Users:

5. In a command window, navigate to the folder in which you have downloaded the source code:
   a. Rename *installOnLEP_Medical.txt* file to .bat, the result should be '*installOnLEP_Medical.bat*'
   b. Execute '*installOnLEP_Medical.bat*'
      i. **to be verified** All cp commands need to be changed to copy/xcopy.
      ii. The if-else statement format needs to be changed to .bat equivalent.

all below to be verified

All users:

8. Run the engine
   a. /path/to/install/loc/OnLEPManager-1.0 --config /path/to/config/file
   b. There is a sample config which should already be set up. You just need to change the file paths to your specific install location.
      i. Look for /path/to/install/location/COPD.cfg for the config file.

ii. Look for /path/to/install/location/msgdata/messages_new_format_all.csv.gz for the sample message data used by default.

iii. Look for /path/to/install/location/logs/output1.txt.gz for the default output logs.

9. Quit the engine by typing "quit". Ctrl-C should also work.

Ignore the following - for Tam tracking only:

All cp commands need to be changed to copy/xcopy.

ii. The if-else statement format needs to be changed to .bat equivalent.

C:\LigaData\RTD\RTD\trunk\SampleApplication\Tools\KVInit\src\main\resources

Deploy & Run Notes

Monday, October 13, 2014 2:22 PM

Note: This is Pokuri's ReadMe. Not yet verified.

Included components

1. Zookeeper shipped with Kafka
2. Kafka
3. Ligadata Engine
4. Ligadata Controller
5. Ligadata UI Component

Prerequisite/Assumptions

- No other Zookeeper Service should be running
- No Other Kafka service is running on this node
- Make sure you run the services/scripts under screens, so that if we loose connection we still be running them.
- Make Sure you replace BINPATH with the real path where we copied vega_0.1.0.tar.gz to uncompress and run

Deployment Steps

1. Unpack vega_0.1.0.tar.gz and move to the folder
   cd BINPATH
   tar -zxvf vega_0.1.0.tar.gz
   cd vega_0.1.0
2. Setup Paths & Give permissions for new sh files
   cd BINPATH/vega_0.1.0
   ./SetPaths.sh
   chmod 777 *.sh
3. Generate Sample Data
   cd BINPATH/vega_0.1.0
   ./GenerateSampleData.sh
4. Depoly UI (RTDWS.war & RT.war are in UI folder)
   currently all jar files are included in the war file itself

   project configuration in war file WEB-INF\web.xml :
   - parameter (ZookeeperConnnectionString)
   - for logging: LogFilePath - must be full path like /usr/share/tomcat6/RTDLogs/npario.log (if path is wrong app will still work, only log will not be written), of course the folder must give write permissions to the user running tomcat

   copy RT.war, RTDWS.war to webapps folder of tomcat

Initialize Required Information (Assuming no services running at this moment)

1. Cleanup Locations
   cd BINPATH/vega_0.1.0
   ./Cleanup.sh
2. Start Zookeeper in Separate Session (Make sure we run it in screen)
   cd BINPATH/vega_0.1.0
   ./StartZk.sh
3. Start Kafka in Separate Session (Make sure we run it in screen)
   cd BINPATH/vega_0.1.0
   ./StartKafka.sh

4. Create Queues, ZNodes & Initialize KV Stroes
    cd BINPATH/vega_0.1.0
    ./CreateQueues.sh
    ./CreateZNodes.sh
    ./InitKvStores.sh
5. Stop Kafka service started in step step 3 using CTRL + C
6. Stop Zookeeper service started in step step 2 using CTRL + C

Steps to Start Services (Assuming no services running at this moment)

1. Start Zookeeper in Separate Session (Make sure we run it in screen)
    cd BINPATH/vega_0.1.0
    ./StartZk.sh
2. Start Kafka in Separate Session (Make sure we run it in screen)
    cd BINPATH/vega_0.1.0
    ./StartKafka.sh
3. Start Engine in Separate Session (Make sure we run it in screen). Wait until we get =>
    cd BINPATH/vega_0.1.0
    ./StartEngine.sh
4. Start Controller in Separate Session (Make sure we run it in screen)
    cd BINPATH/vega_0.1.0
    ./StartController.sh

Connect to UI and START & STOP processing data

Stop Services which we started. All services are running in sessions. we get sessons and do CTRL + C in the following order.

1. Stop Controller
2. Stop Engine
3. Stop Kafka
4. Stop Zookeeper

Install - Metadata.API

Thursday, October 9, 2014 9:51 AM

Note: This is Ramana's guide. Not yet integrated into full process.

Overview

This guide will help you get the RTD Metadata.API installed and ready to use.

The Metadata.API defines the CRUD (create, read, update, delete) operations on metadata objects supported by this system. The metadata objects include Types, Functions, Concepts, Derived Concepts, MessageDefinitions, and Model Definitions. All functions take String values as input in XML or JSON Format and return a JSON string of ApiResult objectd.

Prerequisites:

- Install Scala 2.10.4
- Install the Java Development Kit (JDK 1.7.1 or greater)
- Install joda-convert-1.6.jar into a local directory (example: $HOME/libs). This jar file is needed in the CLASSPATH variable for successful compilation of scala code generated by model compiler.
- The definition of the data models and rule sets used in your business that need to be created.

After you have installed Scala, the JDK, and the RTD engine binary, you can:

- Write your own functions
- Use the JAR to compile those functions

Usage:

Note: these steps were verified only on a Linux(centos 6.5) VM.

NOTE: is this right?

1. In the trunk, (github/RTD/trunk) run the following commands:
   sbt package
   sbt
   > project MetadataAPI
   > assembly

2. Update the config file "RTD/trunk/MetadataAPI/src/main/resources/MetadataAPI.properties" to suit your environment.

3. Make sure the directory specified by the property "JAR_TARGET_DIR" exists. (This directory is a staging directory to compile scala code generated by model compiler.)

Note: CLASSPATH may contain lot of unnecessary jar files, but some of them are required to compile scala code generated by model compiler.

4. Start MetadataAPI test harness using fat-jar java -jar:
   RTD/trunk/MetadataAPI/target/scala-2.10/MetadataAPI-1.0

You will see the following UI. It should be pretty intuitive from here.

<need screenshot of UI here>

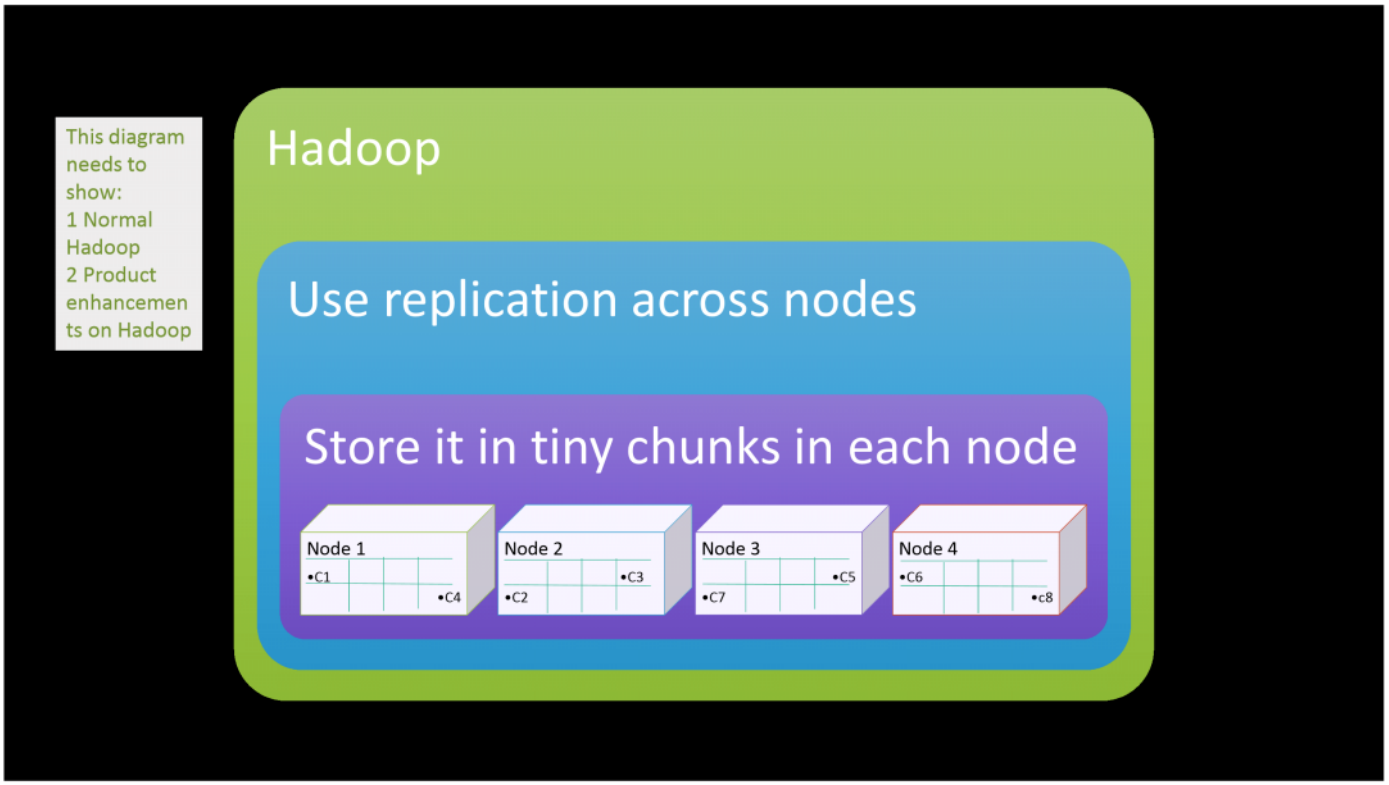
Drawings 20141024

Monday, October 27, 2014 10:33 AM

Drawings 20141024

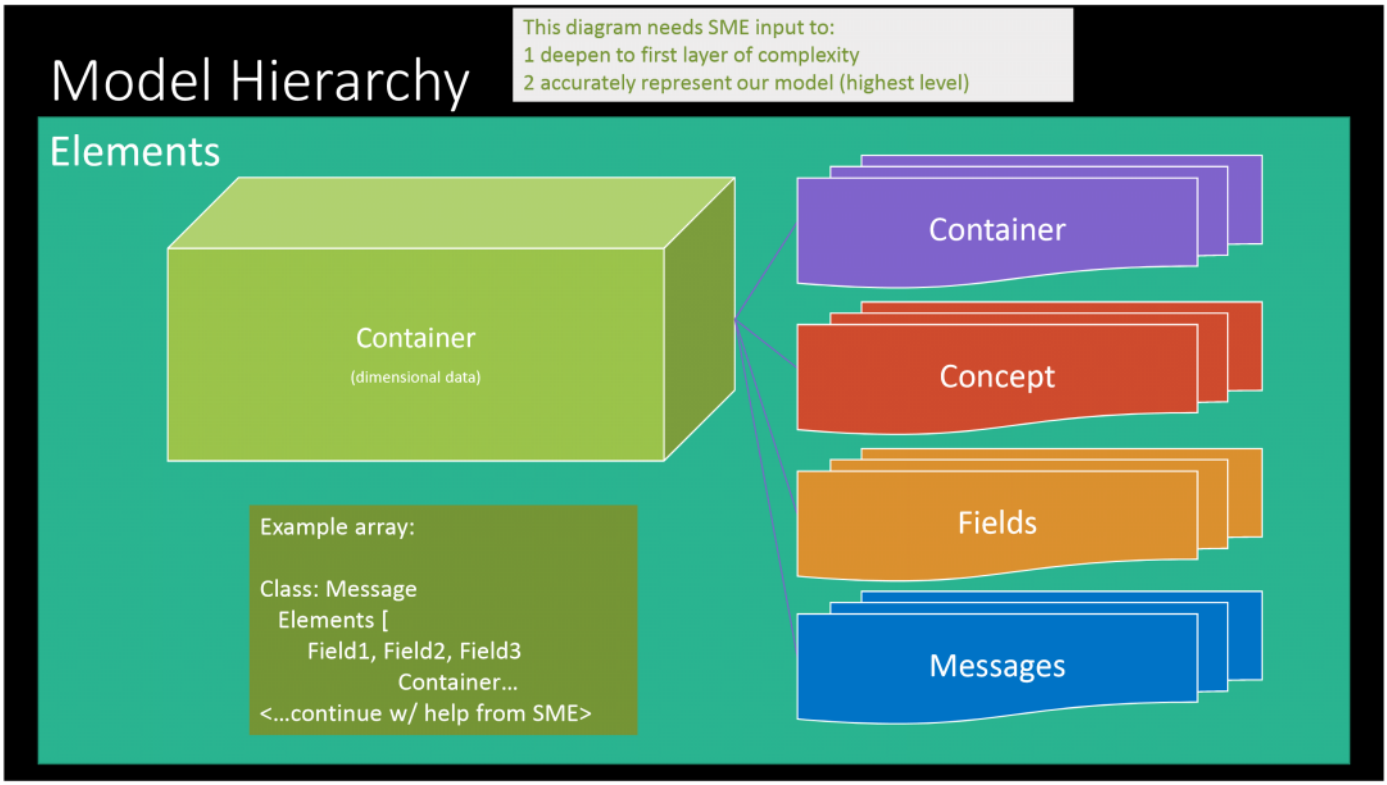
Page 2

Monday, October 27, 2014 10:33 AM

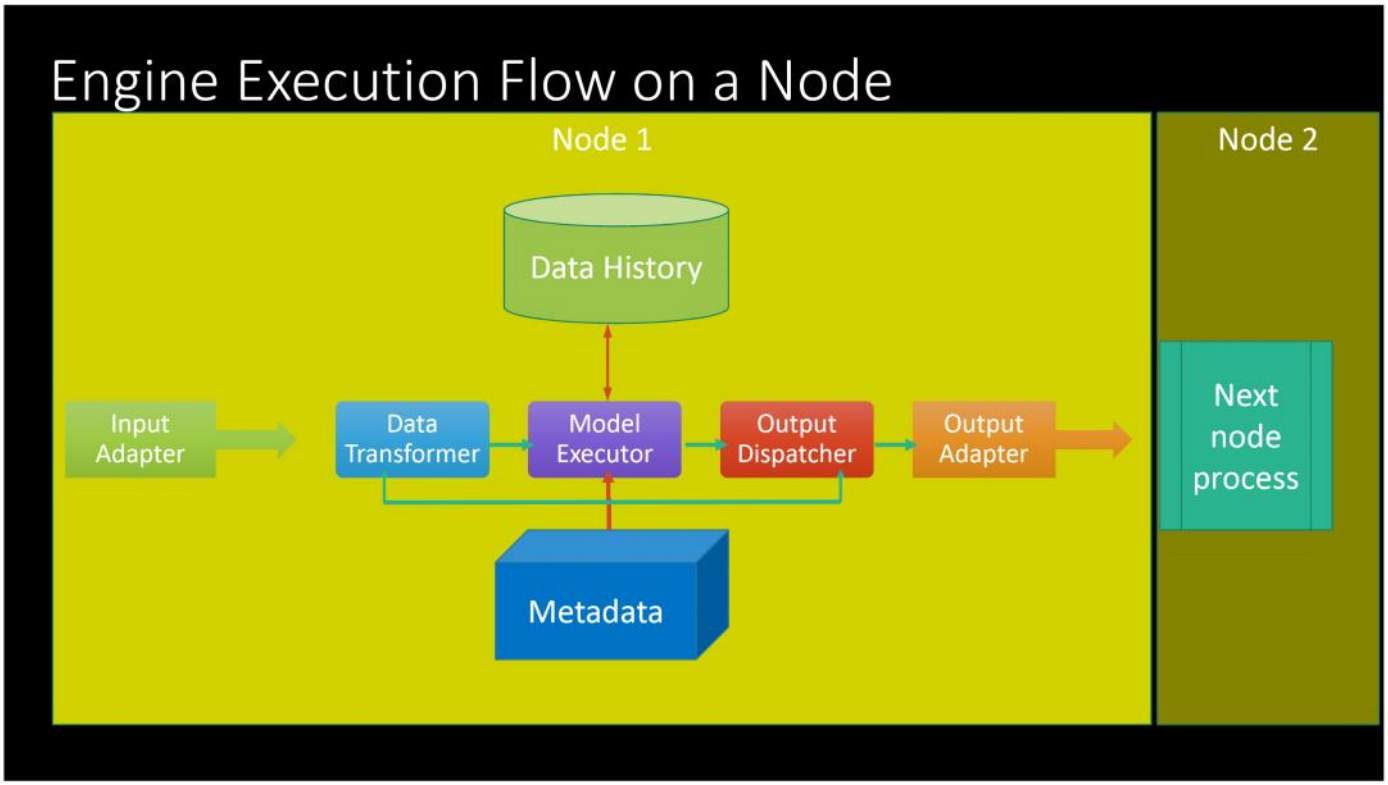


Page 3

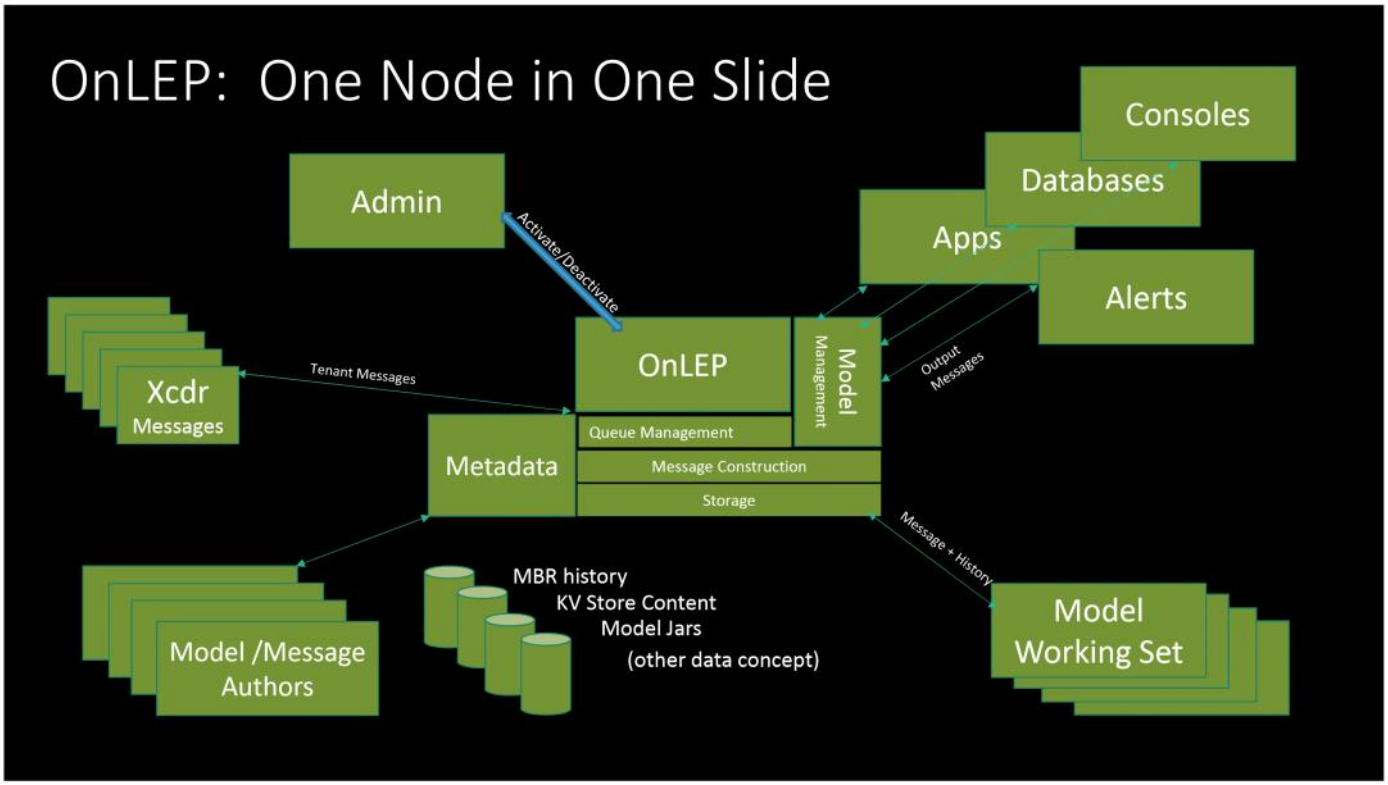
Monday, October 27, 2014 10:33 AM



Monday, October 27, 2014 10:33 AM



Monday, October 27, 2014 10:33 AM



Page 6

Monday, October 27, 2014 10:33 AM

NEXT: