PODIUM DATA SOURCE

OVERVIEW

Podium Data Inc.'s flagship solution ("Podium") delivers a scalable and cost effective Data Lake Management platform that enables users to source, validate, profile, transform and provision data in a self-service environment. There is an undeniable shift in the way enterprises are implementing data solutions. Most are forgoing the lengthy process of building a warehouse that attempts to meet everyone's needs – a long and expensive approach that commonly falls short in meeting original business objectives.

Enterprises are moving towards an agile 'Fit for Purpose' data methodology. This approach provides our customers with data that is business-ready faster and more affordably. Podium, built from the ground up and leveraging proven open source technologies has brought to market a unique platform. This platform manages the entire 'first mile' of data and beyond. Podium has built critical functionality that empowers the enterprise and users by instilling confidence in the data. This is done by combining automated data ingestion, validation and profiling with enterprise level transformation capabilities all from a centralized, easy to use interface. Additionally, as a platform Podium maintains visibility throughout the data lifecycle—for example, tracking detailed ingest statistics and custom built data transformations through lineage.

This document will provide a detailed description of the technical architecture components of Podium data source management functionality and execution behavior.

DATA SOURCE MANAGEMENT MODULE

Podium Data Management enables Administrator Level users to deploy and manipulate enterprise data assets across clustered nodes with ease and efficiency through a powerful ingest framework. The Data Source Administration module enables users to configure Data Sources and Entities in Podium. A Data Source is defined as a collection of Entities, which has physical properties about the sources of those entities (such as LOCALFILE, MAINFRAME, RDBMS) while an entity inherits the source properties and provides additional details such as type of data (FIXED LENGTH, DELIMITTED, CHARCTERSET etc.) The "Add Source" wizard walks a user through adding new sources and/or entities into the system. Once the new source or entity has been defined, the actual data can then be loaded through the LOAD DATA screen. All results of the load are available through the LOAD DATA log screen.

DIFFERENTIATORS

PODIUM IS A PLATFORM

As a platform, Podium is architected to manage and curate all data. The economies of scale of the Hadoop stack, coupled with new levels of precision and control, enable organizations to manage all data from various sources in many formats with unified processes. This approach minimizes the need for multiple disjointed technologies to accomplish similar objectives. The subtle distinction between a tool and platform has major implications in practice: 1) all data is contained, managed, and accounted for within the platform as opposed to 'transient' as with most data preparation and management tools, 2) all ingested data is tracked in detail, and 3) all data transformed within Podium is available on the platform for others to leverage.

ADVANCED DATA INGESTION

Legacy data, regardless of source, needs to be imported and not rejected (as a database will do) and not consumed and processed with no indication that the imported data might be significantly dirty, as many HDFS BI applications will do. Podium ingests data quickly and accurately, categorizing and partitioning data as good, bad, and ugly records (see below for details). These records, in other systems, will generate misleading results with no indication that the data is bad or that the data was improperly parsed.

FIELD-LEVEL VISIBILITY

Podium provides business users with a thorough understanding of enterprise data through comprehensive validation and profiling. On the Podium Platform you can search for data, tag elements at a source, entity, and field level to easily find relevant data for one-off, discrete and/or ongoing analytics projects. The Data Source module provides information including field cardinality, the number of null values in a table, min/max values, and frequency distributions on all fields in the data lake. Podium processes every value in every field from every loaded file automatically, in parallel with Hadoop.

SECURITY

The Data Source Module is only accessible by Users with Admin Level permissions. Podium is sensitive to the importance of data security and addresses application and data security with various measures at both levels to protect individual privacy and enterprise data assets.

User and data security setup are executed through the Podium User Interface within the Security Module. Podium administrates user permissions with groups and roles, with the assignment of data resources to groups allowing individuals to co-exist on the same platform with federated access and field-level granularity to role-specific data.

Rigorous protections are in place to secure application data. These measures include:

- Fine level of object security including Podium Sources, Entities and Fields
- AD Authentication (Kerberos)
- Data Encryption at a field level
- Data Obfuscation based on “sensitive” field definition
- SSL Support via standard web server configuration
- Integrates seamlessly with filesystem level encryption
- All associated passwords are stored as encrypted text

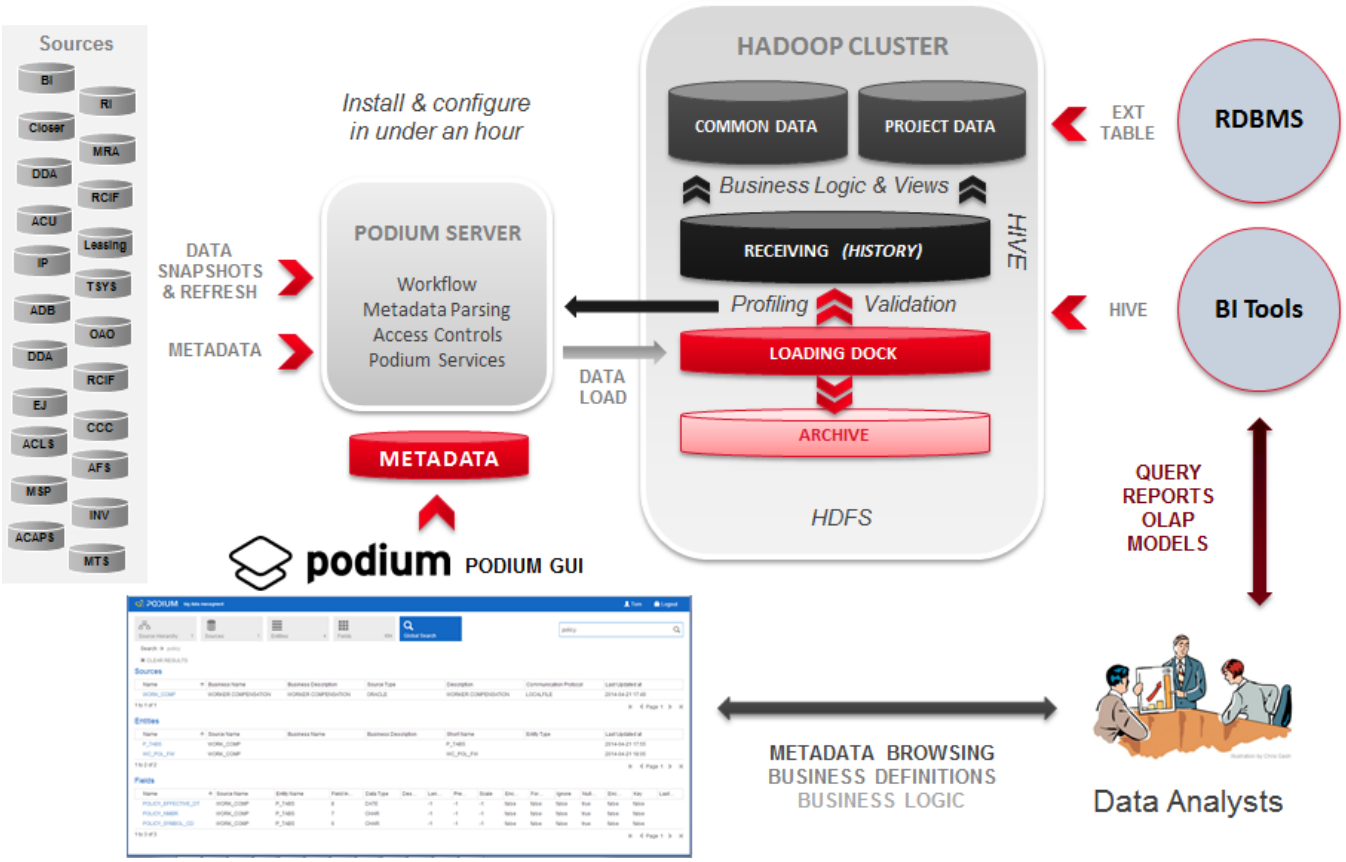
DATA VALIDATION AND PROFILING

The Podium User Interface provides an intuitive way to ingest, manage and transform data. Once data has been loaded via JDBC API or FDL and formatted to HDFS (when not sourced from HDFS) the loading utility performs many validation tests, calculates profile statistics for every field, and registers the data in HCATALOG. The Podium Platform enables business users to validate and govern data that has up until now not been qualifiable as good data. Upon import, the Data Source Module provides a breakdown of the processed records:

Record Status	Description
Good	Validated as having correct record structures and no issues with data type violations or problems with the contents of the data
Bad	Do not conform to specified record layout, such as having the correct number of columns, delimiters, headers and trailers. The most common reasons for bad records are: • Incorrect layout information (wrong fixed length bytes or delimiter) • Embedded delimiters within fields – such as a common in the middle of a description or address field. • Non-ASCII or control characters causing record breaks
Ugly	Match the record format but some of the field data is problematic. Most common reasons for ugly records are: • Data type inconsistencies (e.g., non-numeric data in a field defined as numeric) • Control characters in a field (these cause issues within the Hadoop code stack)

ARCHITECTURE & COMPONENTS

PODIUM DATAFLOW ARCHITECTURE



COMPONENTS

Component	Details
Data Ingest	Enables a user to create the technical metadata about a given source of data through a wizard interface.
Data Ingest	Reads a source of data, performs any data conversions needed, validates, profiles and writes data to HDFS via standard low level HADOOP API calls. Canonical record format: UTF-8, TAB separated.
Character Set Conversion	"on the wire" conversion of character sets such as EBCDIC, LATIN_1 etc. to UTF-8
Ingest Logging	Provides the user with detailed logging information regarding the ingest results including record count
History Management	Handling of history on ingests. Support for incremental and snapshot ingests. All underlying partition management is automated.
Source/Data Support	Cobol Copybook Defined Mainframe (EBCDIC / Binary) data, RDBMS direct connection, Fixed Flat File, Delimited Flat File, Optionally Enclosed fields, Headers, Trailers, XML
Data Typing	Strong explicit data typing including numeric, string, date, timestamp, boolean and binary types
Data Prepare	Enterprise class data transformation capabilities and support.
Tagging	Allows users to create custom tags (metadata) for any object within Podium
Business/Technical Descriptions	Allows users to import and manually enter metadata descriptions at a source, entity and field level
Search	Search for Podium Objects and profile data results. Case insensitive, fuzzy search.

Shop For Data	Feature supporting selection of "data" for export and/or custom data view creation
Field Level Security	Enables administrator to setup field level access to groups / users
Field Level Encryption	AES_128_CBC, shared key, no salt
Password Encryption	Id using Local Authentication, or JDBC source authentication, podium stores passwords as AES_128_CBC, shared key, no salt
Field Level Obfuscation	Ability to, at a field level, select fields for "Obfuscation" Obfuscation types include: Masking, Dictionary First Name, Dictionary Last Name and Random
Integration with File System Level Encryption	Podium integrates with the Hadoop Distributed File System (HDFS) with standard low level HADOOP API calls. Since we read and write data via these calls, we integrate seamlessly to the standard file system level encryption techniques.

DATA SOURCES

SUPPORTED DATA SOURCE SYSTEMS

Connection Type	Source Type	Communication Protocol
FIELD DEFINITION LOADER	FILE SQLSERVER ORACLE MAINFRAME/COBOL HIVE TERADATA POSTGRESQL IBM DB2 MYSQL	LOCALFILE FTP AWS S3 HDFS
JDBC	SQL SERVER ORACLE HIVE TERADATA POSTGRESQL MYSQL	JDBC API

DATA SOURCE DETAILS

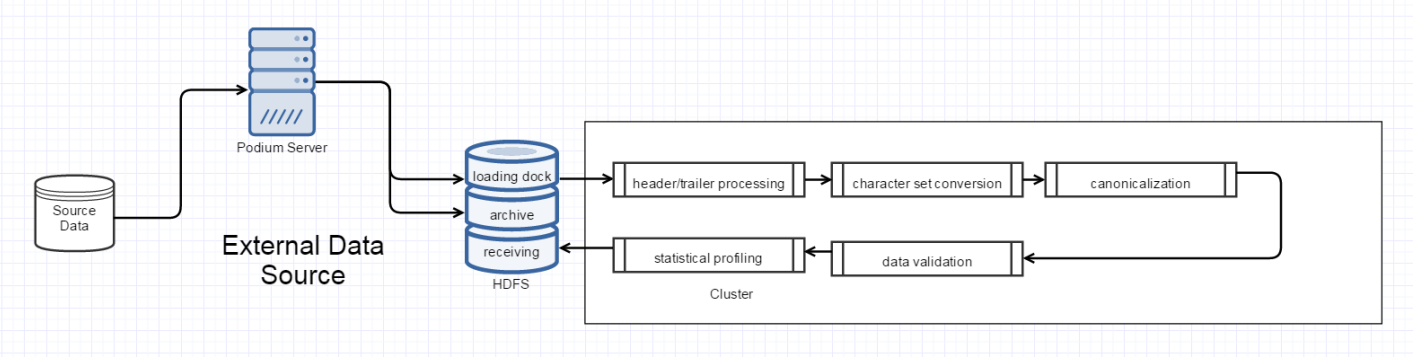
Data Set Type	Details
Mainframe	Cobol Copybook Defined Mainframe (EBCDIC / Binary) data. Includes support for REDEFINES and OCCURS.
Fixed Length	Flat file with fixed byte or character lengths, terminated or non-terminated
Delimited	Flat file with fields separated by a single or multi character delimiter, includes support for optionally enclosed (by quotes) fields where embedded delimiters may occur
XML	XML data sets. Optionally normalized or flattened. XSD optional.
RDBMS	Native connection support for Teradata, Oracle, Postgres and SQLServer via JDBC.
S3	Direct URL support for Datasets on Amazon S3 for all data set types supported by Podium,
SFTP/FTP	Direct connection support for FTP protocols.
HDFS	Direct URI connection to HDFS data files, read and written via HADOOP API
Headers	Support for defining file headers in line or byte count. Will be stripped off as a part of the conversion process.
Trailers	Support for defining file trailers which will be stripped as a part of the data ingest process.
Header/Trailer REGEXP	User can define a regular expression to parse values from headers or trailers such as record counts.
Character Set Type Support	EBCDIC, US_ASCII, LATIN_1, UTF-8,16,32 (LE and BE)

DATA INGEST

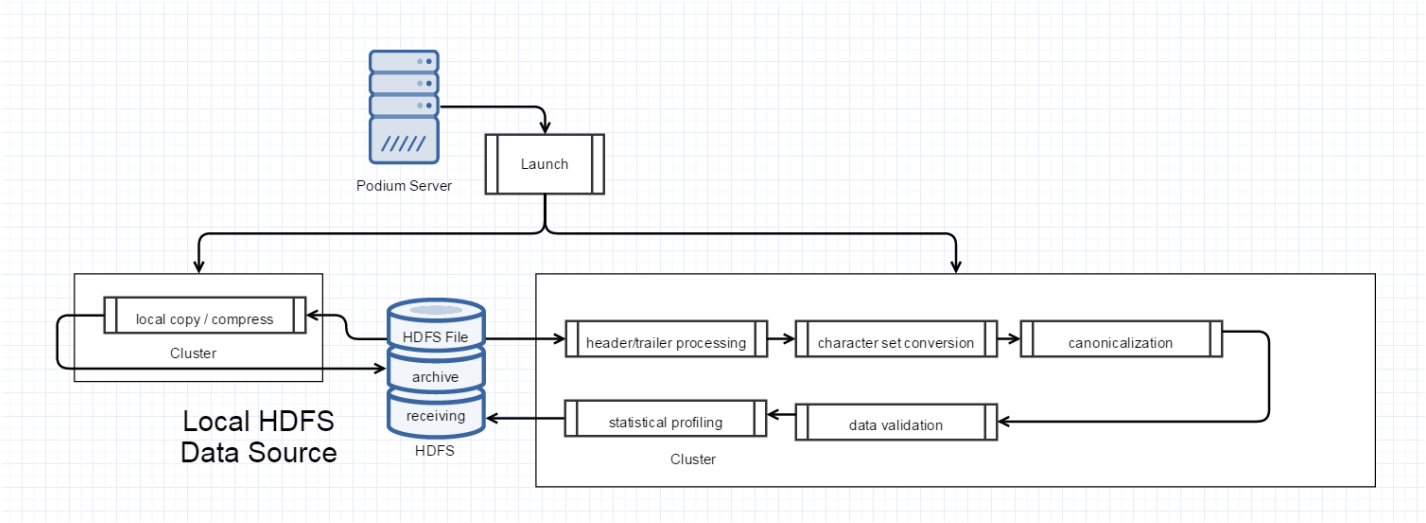
Podium's ingest process calls the "dock hand", "receiving agent" and "distribution" services which performs several distinct functions. These functions are detailed in the table below:

Function	Podium Service	Details
Move to Loading Dock in HDFS	Dock Hand	Writes data into HDFS as a first step (when data is not already in HDFS).
Data Archival	Dock Hand	Compresses and writes data into HDFS in exact source format.
Header/Trailer Processing	Receiving Agent	Based on user defined properties, processes header and trailer information.
Characterset Conversion	Receiving Agent	Read and convert input characterset.
Canonicalization	Receiving Agent	Produces a UTF-8, TSV file in HDFS.
Data Validation	Receiving Agent	Performs record level and field level data validations.
Statistical Profiling	Receiving Agent	Performs field level profiling such as min, max and distribution of values.
History Management	Distribution Agent	Manages snapshot and incremental loads via partition administration in Hive & HCat.
Data Exposure	Distribution Agent	Exposes data for Hive, Impala and PIG by applying structure via "table creates".

EXTERNAL DATA SOURCE INGESTION



LOCAL HDFS DATA SOURCE INGESTION



DIRECTORY DEFINITIONS

As Podium ingests and writes data into the HADOOP Cluster, data is organized as described below.

Directory Definitions
$podiumbase – Root directory (/) which is configurable, and can be multiple levels such as /podium/dev
$sourcename – Name of the "source" as defined in Podium
$entityname – Name of the "entity" as defined in Podium
$partition – Partition directory. Value is based on "current system time" by default, can be overridden by an entity level property.

The following directories are created and managed by Podium.

Dataset Condition	**Written to Podium Directory:**
Compressed "as is source format" datasets for every ingest by default (can be turned off)	*/$podiumbase/archive/$sourcename/$entityname/$partition*
Intermediate, temporary write of source file. Written in parallel with archive write. Will not be used if data source is HDFS.	*/$podiumbase/loadingdock/$sourcename/$entityname/$partition*
All records validated as GOOD	*/$podiumbase/receiving/$sourcename/$entityname/$partition/good*
All records validated as BAD	*/$podiumbase/receiving/$sourcename/$entityname/$partition/bad*
All records validated as UGLY	*/$podiumbase/receiving/$sourcename/$entityname/$partition/ugly*
All detailed profiling results are stored here. This is the combined profile data from the splitprofile directory.	*/$podiumbase/receiving/$sourcename/$entityname/$partition/profile*
Detailed ingest log	*/$podiumbase/receiving/$sourcename/$entityname/$partition/log*

APPENDIX

SOURCE PROPERTIES

Podium Property	Description	Values
src.comm.protocol	communications protocol used to pull data from source	S3, LOCALFILE, HDFS, FTP
src.sys.type	source system type	FILE, SQLSERVER, ORACLE, MAINFRAME, PODIUM_INTERNAL, HIVE, TERADATA, POSTGRESQL, DB2, MYSQL
src.entity.name	name of entity/table	user specified
src.file.glob	wild card glob of data files for this source (or select statement for jdbc source)	User defined (JDBC - system defined, user modifiable)
unicode.byte.order.mark	Unicode byte order mark	UTF_32LE_BOM, UTF_32BE_BOM, UTF_8_BOM, UTF_16LE_BOM, UTF_16BE_BOM
unicode.byte.order.mark.incidence	defines expected BOM incidence	NEVER, SOMETIMES, ALWAYS
header.byte.count	optional integer that specifies the fixed number of bytes in the file header	user specified
header.line.count	optional integer that specifies a fixed number of lines in the file header	user specified
header.defines.field.names	boolean which indicates whether or not the header specifies the field names	TRUE / FALSE
header.record.count.regex	optional regular expression with a capturing group for extracting the record count from the header	Regular Expression
header.validation.pattern.is.regex	boolean flag which controls whether or not header validation pattern is a regular expression	TRUE / FALSE
header.validation.pattern	pattern for validating header ... either exact string match or regular expression	user specified
trailer.byte.count	optional integer that specifies a fixed number of bytes in the file trailer	user specified
trailer.line.count	optional integer that specifies a fixed number of lines in the file header ... where line is terminated by \n or \r\n	user specified
trailer.record.count.regex	optional regular expression with a capturing group for extracting the record count from the trailer	Regular Expression
trailer.validation.pattern.is.regex	boolean flag which controls whether or not trailer validation pattern is a regular expression	TRUE / FALSE
trailer.validation.pattern	pattern for validating trailer ... either exact string match or regular expression	user specified
trailer.control.z.at.eof.incidence	incidence of Control Z/0x1A MS-DOS end of text file indicator	NEVER, SOMETIMES, ALWAYS
record.characterset	character set definition	US_ASCII, LATIN_1, UTF_8, UTF_16LE, UTF_16BE, UTF_32LE, UTF_32BE, US_EBCDIC
record.charencoding.confidence	confidence in the char encoding	decimal in the range 0.0-1.0

Podium Property	Description	Values
record.layout	defined how the record is laid out (fixed, delimited, etc.)	FIXED_LENGTH, FIXED_LENGTH_TERMINATED, VARIABLE_LENGTH_TERMINATED, MAINFRAME_VARIABLE_BLOCKED
record.fixed.byte.count	fixed record byte length for FIXED_LENGTH and FIXED_LENGTH_TERMINATED records	user defined (CCB - utility will determine)
record.min.byte.count	minimum valid record byte length	user defined
record.max.byte.count	maximum valid record byte length	user defined
record.record.terminator	record terminator char sequence for FIXED_LENGTH_TERMINATED and VARIABLE_LENGTH_TERMINATED	user defined
record.field.delimiter	char sequence for delimiting fields for delimited record types	user defined
record.last.field.has.delimiter	boolean indicating whether there is a final field delimiter in the record	TRUE / FALSE
record.validation.regex	optional regular expression used to validate a record	Regular Expression
record.open.quote	char sequence used for opening an enclosed/quoted field	user defined
record.close.quote	char sequence used for closing an enclosed/quoted field	user defined
default.field.embedded.enclosure.scheme	how to deal with an enclosure inside an enclosed/quoted field	NONE or DOUBLE_EMBEDDED_ENCLOSURE
default.field.enclosure.incidence	incidence of enclosed fields	NEVER, SOMETIMES, ALWAYS
default.field.nullif.pattern.is.regex	boolean flag which controls whether or not nullif pattern is a regular expression	TRUE / FALSE
default.field.nullif.pattern	char sequence which represents the NULL value ... exact match or regular expression	user defined
default.field.trim.left	boolean controlling whether or not whitespace should be trimmed from left of fields	TRUE / FALSE
default.field.trim.right	boolean controlling whether or not whitespace should be trimmed from right of fields	TRUE / FALSE
default.field.allow.whitespace	boolean controlling whether or not embedded whitespace is allowed in fields	TRUE / FALSE
default.field.allow.control.chars	boolean controlling whether or not ASCII/Unicode control chars 0x00-0x1F + 0x7F are allowed in fields	TRUE / FALSE
default.field.allow.non.ascii.chars	boolean controlling whether or not non 7-bit ASCII chars are allowed in fields	TRUE / FALSE
default.field.min.legal.char.length	minimum number of chars allowed in a field	user defined
default.field.max.legal.char.length	maximum number of chars allowed in a field	user defined
default.date.format	date format string in Java/JODA date format or vendor-specific date format	user defined
default.min.legal.date	minimum legal date as yyyy-MM-dd ISO 8601 date	user defined
default.max.legal.date	maximum legal date as yyyy-MM-dd ISO 8601 date	user defined

Podium Property	Description	Values
field.trim.left	boolean controlling whether or not whitespace should be trimmed from the left of this specific field	TRUE / FALSE
field.trim.right	boolean controlling whether or not whitespace should be trimmed from the right of this specific field	TRUE / FALSE
field.allow.whitespace	boolean controlling whether or not embedded whitespace is allowed in this specific field	TRUE / FALSE
field.allow.control.chars	boolean controlling whether or not ASCII/Unicode control chars 0x00-0x1F + 0x7F are allowed in this specific field	TRUE / FALSE
field.allow.non.ascii.chars	boolean controlling whether or not non 7-bit ASCII chars are allowed in this specific field	TRUE / FALSE
field.enclosure.incidence	incidence of enclosures for this specific field	NEVER, SOMETIMES, ALWAYS
field.min.legal.char.length	minimum number of chars allowed in this specific field	user defined
field.max.legal.char.length	maximum number of chars allowed in this specific field	user defined
field.min.legal.value	minimum legal value for this specific field ... interpretation depends upon field type	user defined
field.max.legal.value	minimum legal value for this specific field ... interpretation depends upon field type	user defined
field.encryption.type.tag	optional tag to identify the type of animal that is being encrypted	user defined
field.date.format	date format string in Java/JODA date format or vendor-specific date format for this specific field	user defined
enable.profiling	boolean controlling whether or not profiling should be performed on this source/entity/field	TRUE / FALSE
enable.validation	boolean controlling whether or not field validation should be performed	TRUE / FALSE
enable.archiving	whether or not inbound files should be archived	TRUE / FALSE
use.single.receiving.mapper	use a single receiving mapper instead of MapReduce in the cluster	TRUE / FALSE
cobol.copybook	the text of the copybook	System defined
cobol.branch.wiring.instructions	branch wiring instructions to provide guidance for decoding COBOL copybooks with REDEFINES	user defined
cobol.trunc.bin.enabled	COBOL compiler/environment option TRUNC(BIN) for truncation of BINARY/COMP-1 numeric items	TRUE / FALSE
cobol.supports.single.byte.binary	whether or not COBOL compiler supports single byte BINARY/COMP-1 numeric items ... non IBM mainframe	TRUE / FALSE
cobol.sync.alignment.enabled	COBOL compiler/environment option for SYNC word alignment	TRUE / FALSE
cobol.unsigned.packed.decimal.uses.sign.nybble	whether or not the sign nybble holds a digit for unsigned PACKED/COMP-3	TRUE / FALSE
cobol.little.endian	set to TRUE if the source COBOL machine architecture is little endian ... non IBM mainframe	TRUE / FALSE

Podium Property	Description	Values
cobol.allow.overflow.redefines	set to TRUE if you want to allow REDEFINES branches which are larger than the original storage allocation	TRUE / FALSE
cobol.allow.underflow.redefines	whether or not you want to allow REDEFINES branches which are smaller than the original storage allocation	TRUE / FALSE
cobol.numproc.nopfd.enabled	COBOL compiler/environment option NUMPROC(NOPFD) relating to valid/invalid sign nybbles in PACKED/COMP-3 numeric items	TRUE / FALSE
field validation regular expression	special class enabling regular expression checks at a field level	Regular Expression