Contents

[About Ocean's Next Gen ETL Tool (ETLAgent): 2](#_Toc485233965)

[FTL Transformer Configuration 2](#_Toc485233966)

[OCEAN ETL – Schema Mapping Tool 3](#_Toc485233967)

[Message to ETL 3](#_Toc485233968)

[Define Extraction Schema 4](#_Toc485233969)

[ETL Results 7](#_Toc485233970)

[Repeating Group configuration 7](#_Toc485233971)

[FTL Transformer 8](#_Toc485233972)

[Onboarding a New Flow: 8](#_Toc485233973)

[Configurations: 8](#_Toc485233974)

[Steps 10](#_Toc485233975)

[Miscellaneous: 12](#_Toc485233976)

# About Ocean's Next Gen ETL Tool (ETLAgent):

The Enhanced ETLAgent contains the following main components:

Reader

* The Reader is basically an input source provider, where it reads a delimited message from file. Currently two core readers are provided as part of ETL library.
* MultiFileReader : Provides a delimited message by reading multiple files
* XML Stream Reader: Provides a delimited repeating group xml messages by reading huge files eg: of size 8 GB in case of AMC flow.
* Any future new Reader requirement can be easily configured to the ETL with minor changes once the New Reader is implemented.
* The reader considers only the files that are matched with the file name patterns provided in the  etl.properties configuration file.

Transformer

* The Transformer applies the transformation logic on the reader's message. This transformation logic is provided by means of a configuration file called transformerConfig.xml which is generated by using ETL GUI Tool ( [Schema Mapping Tool](http://oceanap01d.nam.nsroot.net:7000/etl/home)[)](http://fxodwap01d.nam.nsroot.net:7000/etl/home). This file contains the table-column data extraction login from the reader's message by using Freemarker Template Language (FTL). Any bad message during this step will be logged in the badrecord files based on etl.properties config (includeBadMessageToLog=true).
* The Transformer instance will be different based on Message type. For example : XML, FIX, JSON, etc.,
* As of now it supports XML, FIX , CSV and JSON only.

Writer

* The writer gets the record from the Transformer via ETL Engine and writes to rollover files which are compatible to the NZ DBloader.

ETL Engine

* ETL Engine is the main component which associates the Reader, Transformer and the Writer. The message is passed via ETL Engines to the other three components. It takes control of these three components and tracks the count of work done by them. The communication among the Reader, Transformer and Writers happens through the ETL Engine.

# FTL Transformer Configuration

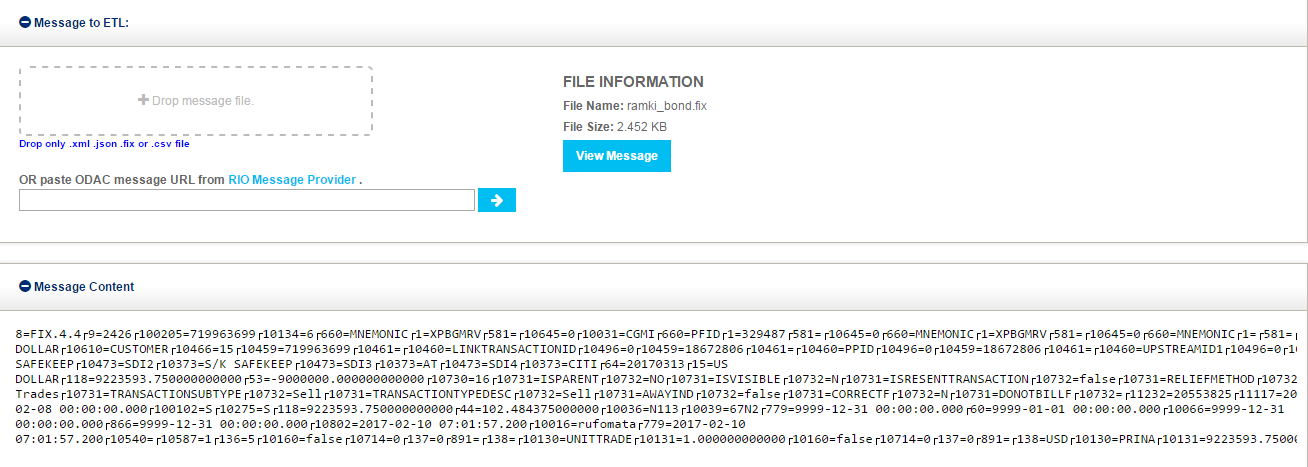
* *Transformer Config file is generated by using the Web application:*[Schema Mapping Tool](http://oceanap01d.nam.nsroot.net:7000/etl/home)
* *Here the user can take the help of GUI to create tables and fields (columns) for each Schema.*
* *Each column holds the actual logic of message transformation by means of Freemarker Template Language (FTL).*
* *This tool can be used to test the FTL logic with sample messages and provides errors (if any) and results in Table Format.*
* *One can test each field/column individually or whole schema which includes all Tables.*
* *It is suggested to have some basic FTL knowledge to write the logic for each column.* [FreeMarker Manual](http://freemarker.org/docs/index.html)
* *There are some utility functions provided internally which can be accessed through Functions link.*
* *Users can drag-n-drop his message file (only .fix,.csv, .xml or .json files) and test the whole schema.*
* *NOTE: Once you download this file from web, open and edit the xml header in the first line, change it to <?xml version="1.0" encoding="UTF-8"?>.This is to avoid some issue with perforce Unicode.*

## OCEAN ETL – Schema Mapping Tool

* *The web application url for the tool is* [Schema Mapping Tool](http://oceanap01d.nam.nsroot.net:7000/etl/home)
* *The tool has 3 sections namely Message to ETL, Define Extraction Schema and ETL Results.*

### Message to ETL

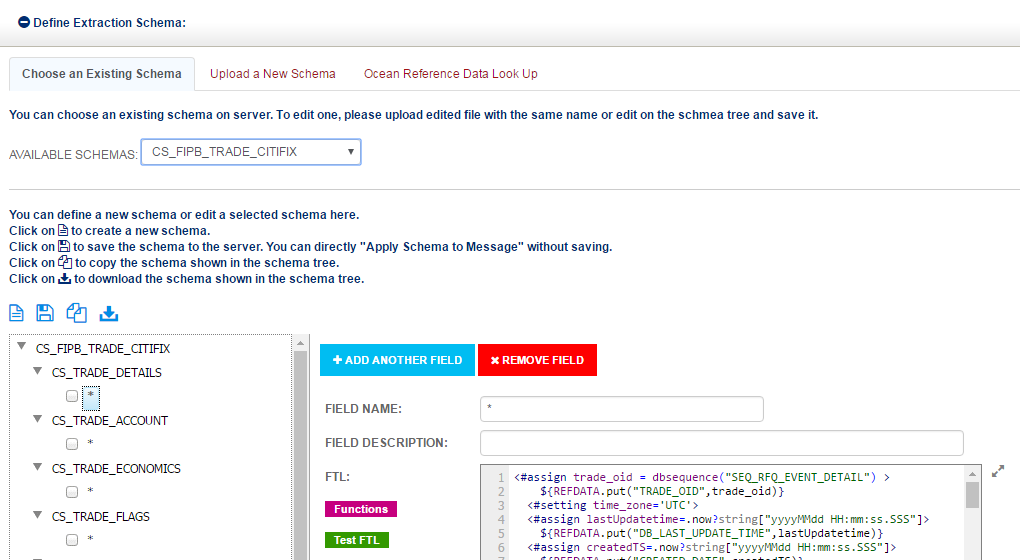
* *This section is used to view the message. User can drag and drop the message less than 1 MB to view the message.*
* *User can drop only .csv,.fix,.xml or .json.*
* *Below is the screen print of the “Message to ETL”*



* Once the user drops the message, User can view the message by clicking the view message *button.*
* *User can view the message under Message Content section and verify the message for its correctness.*

### Define Extraction Schema

* This section is used by the user to define his schema. This section has three tabs which are Choose an existing schema, Upload a new schema and Ocean Reference Data look up.
* Choose an Existing Schema: User can use already existing schema to implement the changes. Also user can create a new schema.
* Upload a new Schema: User can upload new schema by uploading the existing transformerConfig.xml file and rename it as needed.
* Ocean Reference Data Look up: User can look up a value from DEV Ocean Reference Data.
* Below is the screen print of the “Define Extraction Section”



* User can add table by clicking on “+ADD Table”, also can remove table by clicking on “xRemove Table” .
* User can add fields by clicking on “+ADD Another Fiels”, Also can remove the field by clicking on “xRemove Field”.
* All the table names and field names must be same as the objects in the database, else the data load will fail.
* FTL code should be written in the FTL template text box. FTL can be written for each individual field or for all fields together using “\*” in the Field name.
* User can verify/check his FTL code for syntax errors and other errors by clicking on Test FTL.
* Repeating group for FIX files should be mentioned at schema level as shown in below screen shot. The message can be viewed in json format by clicking on the “convert FIX to JSON” .



FTL code sample snippet for FIX files



Doc[“75”] captures tag 75 value.

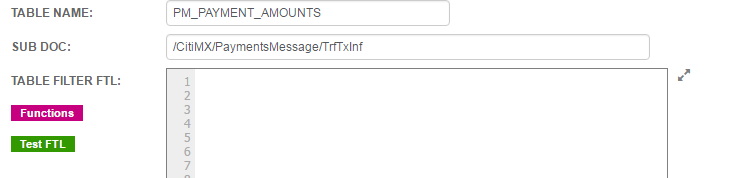
FTL code sample snippet for .csv files



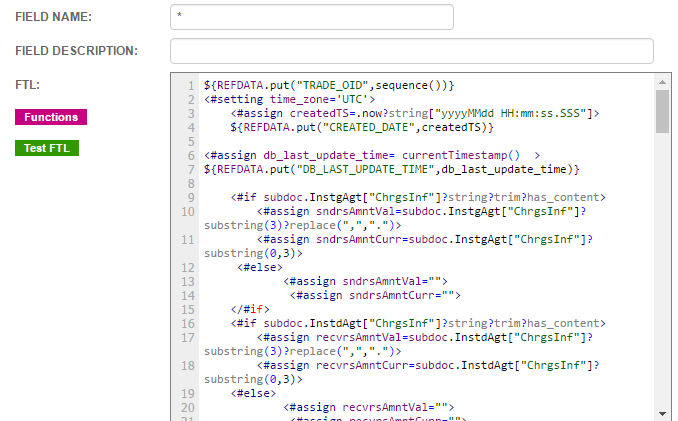
Doc[“10”] captures value of field 11(position) separated by a delimiter.

FTL code sample snippet for .xml files

AT table level



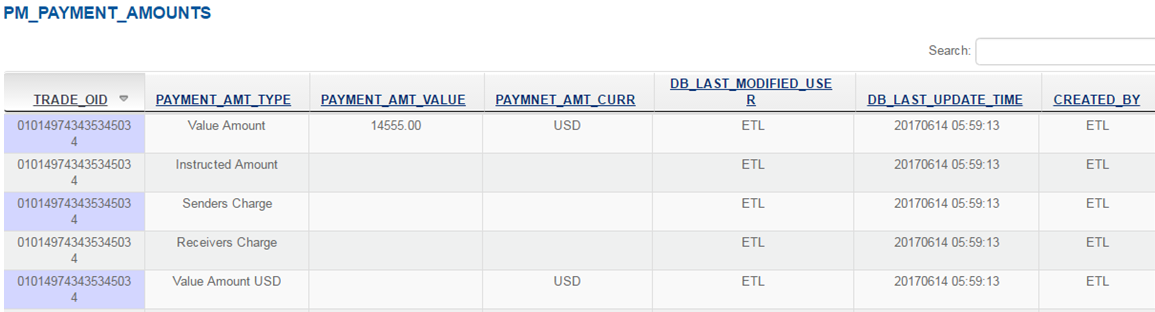
At field level



Subdoc.InstgAgt[“ChrgsInf”] captures the tag value of /CitiMX/PaymentsMessage/TrfTxInf/InstgAgt/ChrgsInf.

### ETL Results

* This section is to test the code developed and check the output of the FTL. User has to click “APPLY SCHEMA TO MESSAGE” to see the results/output.
* Below is the screen print for ETL Results



### Repeating Group configuration

* Fix file have repeating tags and to handle those tags we need to configure the repeating group file.
* The File consists of OBJ\_NM#NAME#COUNTTAG#MEMBERTAG#ROWSTARTTAG#FLOW#PARENTGROUP
* OBJ\_NM : Name of the object
* NAME : Tag Name
* COUNTTAG : Tag that refer the count
* MEMBERTAG : List of Tags for the OJB\_NM
* ROWSTARTTAG : Starting tag for the repeating group
* FLOW: Flow name
* PARENTGROUP: Parent of the group.
* Sample record rg.account#account#10134#660,1,581,10645,10697,10698,10699,10031#660#testFIX#null

# FTL Transformer

* *The FTL Transformer is the core transformation logic which is applied by the Transformer on the message provided by the Reader. The core logic resides inside the Ocean commons library. The main aim of this tool is to avoid introducing new java code specific to each flow.*

# Onboarding a New Flow:

To onboard a New Ocean Flow, Please follow the below **Configurations** and **Steps**.

## Configurations:

* *Every New Ocean Flow/System running under this enhanced ETLAgent will have the below 5 type of configurations.*
* *Most of the work is done in tranformerConfig.xml and this is the place where the message transformation/extraction logic lies.*

*tranformerConfig.xml*

* *Generate this config file by using the* [*Schema Mapping Tool*](http://oceanap01d.nam.nsroot.net:7000/etl/home) *as explained in the above FTL Transformer Configuration section*
* *Freemarker Version used : 2.3.21*
* *Sample generated file can be found at perforce:*
* */global/ocean/data/ocean/ocean\_etl/current/working/inst.glob\_csl\_to\_ocean\_etlagent\_1/config/transformerConfig.xml*
* *This file allows user to apply some logic during the message transformation specific to individual column.*
* *NOTE: Once you download this file from web, open and edit the xml header in the first line, change it to <?xml version="1.0" encoding="UTF-8"?>*



*etl.properties*

* *This properties file contains all the key-value properties to be used inside the library to control the logic, decision making, resource selection, location configurations, etc..*
* *Some entries:*
* *inputFilePattern : To file patterns for the reader to read the matched files*
* *inputBaseDirectory: The base ocean directory for this flow to read input data files.*
* *handleInvalidXMLMessage :  a boolean config to handle bad ocean xml message. Default to false.*
* *includeBadMessageErrorToLog :  a boolean config to decide whether to include the error message to the badrecords data file. Default to false.*
* *includeBadMessageToLog : a boolean config to decide whether to include the message itself to the badrecods data file. Defaults to false. This is useful for the flows like AMC where the input file is huge in size and complete data in single line.*
* *Sample file in perforce: /global/ocean/data/ocean/ocean\_etl/current/working/inst.glob\_csl\_to\_ocean\_etlagent\_1/config/etl.properties*



*applicationContext.xml*

* *This config file is the Spring application context file used to create and manage the reader, transformer, writer and ETL Engine objects.*
* *Any of these new components introduced in future will be configured here, without any code change to the ETL Engine.*
* *This file also has some utility/helper beans configured.*
* *Sample file in perforce: /global/ocean/data/ocean/ocean\_etl/current/working/inst.glob\_csl\_to\_ocean\_etlagent\_1/config/applicationContext.xml*
* *Most of the times this file would be same and reused across different flows. Development team will take care of this.*
* *FIX, XML, JSON, CSV type message will have different transformations in this config file.*



*fixrepeatinggroup.config*

* *FIX Message Repeating Group (Count Tag) Configuration File. Find the sample below.*



*surrogateKeyService.properties :*

* *To generate Surrogate Key from Database using NZ DB sequencer.*

**

*META Data (etlagent.sql,common.sql)*

* *Any Ocean flow requires the DB scripts to configure instance name and other configs in the OceanMeta Database.*
* *Other meta config includes environment, system, business date, jvm args, cache host-port, jmx port, etc.,*
* *It also includes Environment specific hostname and serverPort to connect to Cache services.*
* *etlClientCache.xml have some placeholder like @hostname\_1@ , @server\_port\_1@  whose values are mentioned in etlagent.sql script.*
* *@hostname\_1@ , @server\_port\_1@ count will differ in Env. For DEV is 2, UAT -2 and for PROD there are 5 values.*
* *These config DB scripts will differ for different Environments.*
* *Also consider preparing META for dbloader.sql, recon.sql and other as per your flow requirements*



## Steps

* *Create above all configurations and execute the Meta Data Scripts in the respective Database Env.*
* *Create the log location*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/elt*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/demultiplexer*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/dbloader*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/odac*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/common*
* *mkdir /global/ocean/data/ocean/log/glob/<SYSTEM>/etlagent*
* *Add an entry to perforce's Ocean\OceanEQDW\populateMeta.bash at respective (DEV, INT, UAT, PRD) location in this file, comment the line and submit the code.*
* *LIST="${LIST} ETL|<system>|<system>|<system>|<region>\_\${SYSTEM}\_to\_ocean\_etlagent\_config\_|${ETL\_COUNT:-1}|Y"*
* *Run the populateMeta.bash to setup (upload configs to Database from the latest updated perforce workspace) the new flow by un-commenting the line which you added for this flow.*
* *Test Run the application (DEV or INT Only)*
* *$ > cd /global/ocean/data/ocean/ocean\_etl/current/bin*
* *ksh ETLAgent.ksh -i inst.<region>\_<system>\_to\_ocean\_etlagent\_1 -d 20130502*
* *After the above command one can see the flow's working config setup at /global/ocean/data/ocean/ocean\_etl/current/working/inst.<region>\_<system>\_to\_ocean\_etlagent\_1/config*
* *The above instance name is as per your entry in Meta Data Scripts for your flow*
* *To automate, add this new flow execution to the Autosys as per the exiting PRD etlagent process/steps.*

*Autosys Jobs*

* ***147273\_OBO\_IncBusDate\_GL\_<SYSTEM>*** *: This job is to incement the Business date on daily basis.*
* ***147273\_OBO\_MOVE\_TO\_ODAC\_GL\_<SYSTEM>*** *: This job is to move feed files from incoming folder to the Odac folder.*
* ***147273\_OBO\_ETL\_GL\_<SYSTEM>\_S*** *: This job is to create the clean files from the source feed files.*
* ***147273\_OBO\_DBL\_GL\_<SYSTEM>\_S*** *: This job is to load data to the session and daily tables in the Netezza Database.*
* ***147273\_OBO\_MARTEnrich\_GL\_<SYSTEM>*** *: This job is to Enrich data in the daily Data Mart table.*
* ***147273\_OBO\_COMDayRpt\_GL\_<SYSTEM>*** *: This job is to generate the recon report.*
* ***147273\_OBO\_ELTLoad\_GL\_<SYSTEM>*** *: This job is to load data from Daily tables to the fact tables.*
* ***147273\_OBO\_ETL\_GL\_<SYSTEM>\_E*** *: This job is to end the 147273\_OBO\_ETL\_GL\_<SYSTEM>\_S.*
* ***147273\_OBO\_DBL\_GL\_<SYSTEM>\_E*** *: This job is to end the 147273\_OBO\_DBL\_GL\_<SYSTEM>\_S.*

## Miscellaneous:

* *ETL Server Location:  /global/ocean/data/ocean/ocean\_etl/*
* *ETL Application working configs : /global/ocean/data/ocean/ocean\_etl/current/working/inst.<region>\_<system>\_to\_ocean\_etlagent\_1/config*
* *ETL GUI Tool Server Location : /global/ocean/data/ocean/ocean\_etltool/*
* *Logs Location : /global/ocean/data/ocean/log/*
* *Ocean Data Location: /global/ocean/data/ocean/data/<region>/<system>/etlagent*
* *Bad Messages logging : The messages generating exception during transformation phase are considered as bad messages.*
* *It will generate a .bad file for ODAC data processing.*
* *Now .bad files default content is :*
* *< msg processed timestamp> ^A <odac file path having this bad record> ^A <bad record position(coutner) in ODAC file>^B*
* *If you add includeBadMessageErrorToLog=true to  etl.properties, it will add error details causing this bad message to .bad file as :*
* *<msg processed timestamp> ^A <odac file path having this bad record> ^A <bad record position(counter) in ODAC file> ^A  < error message> ^B*
* *If you add includeBadMessageToLog=true along with above config to  etl.properties, it will include odac message itself to .bad file as :*
* *<msg processed timestamp> ^A <odac file path having this bad record> ^A <bad record position(counter) in ODAC file> ^A  <error message> ^A  <bad odac  mesage> ^B*
* *The above configs would make the Support and Dev teams’ life easy to debug the issue.*
* *If you add  includeOnlyBadMessageToLog=true (In this case it will simple ignore the other two properties: includeBadMessageErrorToLog, includeBadMessageToLog,           even though they hold true.)*
* *<bad odac mesage> ^B*