

|  |
| --- |
| Telavance, Inc. |
| *WEI™ Design Document* |
| Version 1.0 |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 12/04/2010 | DRAFT | First DRAFT | Rukmini Pappu |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Document Purpose 4

2. System Architecture Overview 4

2.1 Development Environment 4

2.2 Deployment Environment 4

2.3 Design Details 4

2.3.1 Message Formats 4

2.3.2 Configuration 5

2.3.3 CTC Detection Rules 6

2.3.4 Message Flow 7

2.3.5 Components 9

2.3.5.1 Wei service 9

2.3.5.2 Wei Translation Service 10

2.3.5.3 Prime Handler 11

2.3.5.4 File Driver 11

2.3.5.5 MQ Driver 11

2.3.5.6 Wei Dashboard 11

2.3.6 Database changes 12

2.3.7 Logging and Exception Handling 20

2.3.8 Security 20

2.4 Packaging 20

2.5 Limitations 20

# Introduction

Transactions in financial institutions contain Chinese telegraphic codes, which occur as sets of 4 numbers. These codes can occur in party fields like the beneficiary and in instructions. Search engines fail to detect matches on them. An institution’s AML program is not complete because monitoring cannot be done on these fields. WEI™ is a translating engine, which takes in the incoming text and applies rules to detect CTC code in the fields and converts them to English text and send it to any pre-configured destination.

## Document Purpose

This document is created to describe WEI™ product features and the design of the product. The document covers the following topics in detail:

* UI Design
* Database Design
* Workflow
* Class Design

# System Architecture Overview

WEI™ is middleware product developed to accept input messages of any format and check for any CTC codes existence and translate them to corresponding English text and save the translated text in pre-configured destination.

CTC (Chinese Telegraphic Codes) codes are sets of 4 digit numbers that occur in message text. CTC numbers range from 0000-9999.

The translation of CTC codes to English is a two step process; the first step involves translating the 4 digit numbers to corresponding Chinese characters using a translation table. The second step involves translating the Chinese characters to English using translation services API provided by Google or Bing. WEI™ supports both Google and Bing.

Wei will initially have the ability to consume messages for translation from a directory or MQ. Since the translated messages can be consumed from various systems/search engines,Wei has handlers to alter the input source message to alter the input message to the format expected by the destination system.

Messages to be consumed by prime

## Development Environment

Requirements due to third party API usage

Internet connectivity

To be provided by lakshman

SQL Server 2005 and above

.Net Framework 4.0

## Deployment Environment

To be provided by lakshman

SQL Server 2005 and above

.Net Framework 4.0

## Design Details

### Message Formats

Wei product will be able to support file driver and MQ interfaces. The supported message formats are

XML

Free format

SWIFT

The above message formats can come through file driver or MQ interface. Following rules will be used to identify the incoming message format

**SWIFT Message:** If the incoming message begins with “{1:” and is followed by 25 characters and then a “}”, then the message is considered a SWIFT message.

**XML Message:** If the incoming message has a <Data> tag, then the message is considered as an XML Message. This is in accordance with the Prime XML message format.

**FreeFormat Message:** If the message is not SWIFT or XML format, the message is considered freeformat

The following logic will be used for message type identification



### Configuration

For each message, the translator looks for the tags specified in the “app.config” file and extracts the values in the tag for translation.

In the app.config file the following tags need to be added to indicate the portions of message to translate.

<Swift translateAllTags="true">

<Tags>

<Tag name="20"/>

<Tag name="32A"/>

<Tag name="54A"/>

<Tag name="71A"/>

<Tag name="59"/>

<Tag name="70"/>

</Tags>

</Swift>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Value** | **Description** | **Mandatory** | **Default** |
| Tag | Any valid SWIFT tag | Required. Informs the tag name for which the settings should be applied. | Y |  |

For each SWIFT tag that needs to be checked for CTC code, we need to specify a tag in the app.config.

### **CTC Detection Rules**

The rules for identifying the CTC codes in a message will be specified in the service config file. The attribute “ctcDeterminingCount” will specify the number of consecutive 4 digit numbers that will be considered as CTC codes by the service parser.

<Translator currentTranslationProvider="bing" currentLanguage="zh-CHS" ctcDeterminingCount="2" noOfRetries="5" ctcAllowedChars=",.">

Sample text with CTC codes is included here, which will give the personnel involved in the project an idea as to how CTC codes occur in message text. The text highlighted in red in the message is how the CTC codes occur

***Sample swift message with CTC codes***

{1:F21PCBCUS33AXXX1022040429}{4:{177:1010221930}{451:0}}{1:F01PCBCUS33AXXX1022040429}{2:O2021926101022PCBCCNBJXXXX10220404291010221930N}{3:{119:COV}}{4:

:20:GDX05OR1094307

:21:GDX05OR1094307

:32A:101022USD9300,00

:53A:/900000010001

PCBCCNBJ

:57A:BKCHUS33

:58A:BKCHCNBJ

:72:/ACC/SHANG BU SUB BR

//COVER OF MT103

//REFNO:GDX05OR1094307

:50K:/44014208000229114012

0154 1472 4430 3843 2448 1367 1446

0455 0756 2589 7098 0361 0674

7311 1795 0243 2403 0005 3163 1142

1129 2890 0589

:57A:BKCHCNBJ45A

:59:/810900692108092014

1639 2639 5478 5114 6639 0948

7139 6663 6551 0108 3810 2589

7098 0361 0674 3234

1101 0433 0361 0674

:70: 6663 6551 6316 3938

4099 4384 5714 01947510

:72:/ACC/SHANG BU SUB BR

-}

### **Message Flow**

The message flow is as described below



The process flow is as follows

* Interface drivers get a processing request (file for a file driver, message for a MQ driver)
* Interface driver calls the Wei service to record the file/message in a db (for reconciliation and reporting) and hands over the processing to the service
* Wei service calls the configured handler (Prime) to get the real data for the OFAC check
* If it is a swift message, the message is parsed and translated for the configured tags
* If a non-swift message, translates the whole message
* Wei service stores the whole translated message to the database
* Call the configured handler (Prime) to pre-format the message (if any) and then send the information for OFAC check
* Once the interface receives the OFAC response, Wei service starts the processing again
* Prime handler is used to create the response back to the calling application
* Wei service uses the interface driver to reply back to the calling application



### **Components**

Wei product has following components

* Wei service
* Wei translation service
* Prime handler
* File driver
* MQ driver
* Dashboard

Each of the components is described in detail below.

#### **Wei service**

This is a windows service, which receives input messages from single or multiple sources and invokes Wei translation service for translation. The service based on drivers defined in the system can monitor a pre-configured directory or MQ queues. The service picks up the message when a new message comes in directory/queue and translates it.

Wei will have the ability to configure multiple file driver and MQ drivers. The service will spawn an individual thread for every file driver configured. For MQ driver, the service will use a thread from a pool of threads available. The number of threads in the pool will be configurable and specified in the Wei Service configuration file.

In the drivers table in Wei database, there is a record defined for every driver. For example, if there are 2 file driver and 3 MQ drivers then this table will have 5 records, 2 for file driver and 3 for MQ driver. Each will have a configurable parameters associated with them. The configuration file is associated as an attribute of each driver.

Wei will use Microsoft Enterprise Library for connecting to the database. There will be a pool of database connections that the individual drivers will use for logging. The connection to the database will be specified in the config file as shown below:

<dataConfiguration defaultDatabase="WeiDB" />

<connectionStrings>

<add name="WeiDB" connectionString="Data Source=(local);Initial Catalog=Wei;Integrated Security=True;Pooling=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

Every request is assigned a unique request ID and stored in the database. This Id is used for reconciliation across the system. Along with the request id, we will store the name of the file for file driver and the message id for the MQ messages.

#### **Wei Translation Service**

Wei translation service is a self sufficient logical code component that contains all function/methods related to translation of text. This is not a separate service.

Translation is a two step process. The first step involves converting the 4 digit codes into Chinese and converting the resulting text into English.

Translation from 4 digit codes to Chinese will be done based on the mapping table. The 4 digit codes range from 0000-9999. There exists a one to one mapping for each of these numbers to a character in Chinese. A lookup will be done when CTC code is detected and corresponding value is returned. By default the mapping table provided will be for mainland china. The location to the map file will be specified in the Wei service config file:

<MapFiles>

<MapFile description="CTC map for chinese traditional" language="zh-tw" path="C:\Telavance\Development\Wei\code\WeiService\data\Traditional Chinese.ctc.map" keyindex="0" valueindex="1"/>

<MapFile description="CTC map for chinese simplified (bing)" language="zh-CHS" path="C:\Telavance\Development\Wei\code\WeiService\data\simplified Chinese.ctc.map" keyindex="0" valueindex="1"/>

<MapFile description="CTC map for chinese simplified (google)" language="zh-CN" path="C:\Telavance\Development\Wei\code\WeiService\data\simplified Chinese.ctc.map" keyindex="0" valueindex="1"/>

</MapFiles>

Once the text is translated into Chinese, the text is submitted to be converted into English. This translation from Chinese to English is done by a third party API. Wei supports Google or Bing API. The APIs used will be described in detail below in section 2.4.1 and 2.4.2.

Anytime the translation is invoked, if there is a failure, we have a retry mechanism. We try translation for 5 times and if it fails, we them move the message to error directory/queue with the status ERROR\_TRANSLATED. The retry mechanism will be specified in the Wei Service config file as shown below:

Wei will support translation of Chinese Characters to English using either Google or Bing translation providers. The translation provider that Wei should use will be specified in the Wei service config file as shown below:

<Translator currentTranslationProvider="bing" currentLanguage="zh-CHS" ctcDeterminingCount="2" noOfRetries="5" ctcAllowedChars=",.">

The” providers” element in the config file will specify the translation providers that Wei will support

<Providers>

<Provider name="google" className="Telavance.AdvantageSuite.Wei.WeiTranslator.GoogleTranslationProvider" maxLength="5000" version="2.0" key="AIzaSyCMab5swnNDuX1m6jDucig51qnnyszOef0" />

<Provider name="bing" className="Telavance.AdvantageSuite.Wei.WeiTranslator.BingTranslationProvider" version="2.0" key="5F1390DD62482373E10E07DC5F55FDE6E5A78722"/>

</Providers>

#### **Prime Handler**

This component handles processing that is specific to Prime XML messages. This component will be specifically used on messages that will be OFAC checked with PRIME OFAC Reporter.

#### **File Driver**

The file driver component monitors the input directory based on the specification in the input file and logs the status and invokes Wei Translation on the text that needs to be translated.

#### **MQ Driver**

This driver monitors incoming MQ messages and sends them for OFAC check. Responses received from OFAC system will be received by MQ driver.

#### **Wei Dashboard**

This web based dashboard is used by users to monitor the status of the messages. The dashboard displays as to where a message is in the workflow and the users can use it to monitor.

### **Database Changes**

***Database Name:*** Wei

***Table Name:*** Audit Table(change the table name to event)

***Purpose:*** Table to store the message workflow along with the status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for the Audit | Y | N |
| Requestid | Int | Id of the request |  | N |
| Level | Int | Audit Level (Debug -1 , Info -2, Error -3, Fatal 4) |  | N |
| Status | Ntext | Status when the audit was created |  | Y |
| CreateDate | Datetime | Date and time the event was created |  | N |
| createOper | Varchar(20) | Operator code who initiated the action |  | N |

***Stored procedures***

Following stored procedures will be created

* + Wei\_AddAudit
  + Wei\_GetAuditMessages
  + Wei\_GetAuditMessagesByRequest

***Table Name:*** Drivers

***Purpose:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for a driver | Y | N |
| Name | Varchar(255) | Logical Name of the driver |  | N |
| Dll | Varchar(255) | Name of the implementing dll |  | N |
| Type | Varchar(255) | Any config required for this driver |  | Y |
| Config | Text | Unique id for a driver |  | Y |
| CreateDate | Datetime | Date and time the record was created |  | N |
| createOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

***Table Name:*** Handlers

***Purpose:*** This table has a list of handlers that Wei supports (like PRIME)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for a handler | Y | N |
| Name | Varchar(255) | Name of the handler |  | N |
| Dll | Varchar(255) | Name of the implementing dll |  | N |
| Type | Varchar(255) |  |  | Y |
| CreateDate | Datetime | Date and time the record was created |  | N |
| CreateOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

***Table Name:*** Interfaces

***Purpose:*** This table has a list of interfaces that we configured

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for a interface | Y | N |
| Name | Varchar(255) | Logical Name of the interface |  | N |
| Handler | Varchar(255) | Id of the handler |  | N |
| Driver | Varchar(255) | Id of the driver |  | N |
| FileFormat | Varchar(100) |  |  | N |
| Config | Ntext | Unique id for a interface |  | Y |
| CreateDate | Datetime | Date and time the record was created |  | N |
| CreateOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

Following stored procedures will be created

***Stored procedures***

Following stored procedures will be created

* Wei\_GetInterfaces

***Table Name:*** Requests

***Purpose:*** This table has all the requests along with their statuses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for the request | Y | N |
| Name | NVarchar(255) | Logical Name of the request (file name/message id) |  | N |
| InterfaceId | Int | Id of the interface processing this request |  | N |
| MessageBody | Ntext | Unmodified contents of the message |  | N |
| Status | Int | Status of the message |  | N |
| Processlock | Int | Need Description |  | Y |
| TranslatedMessage | Ntext | Message with English translation added to the message |  | Y |
| CreateDate | Datetime | Date and time the record was created |  | N |
| CreateOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

Following stored procedures will be created

***Stored procedures***

Following stored procedures will be created

* + Wei\_AddRequest

Parameters:

Purpose:

* + Wei\_GetMessageStatus

Parameters:

Purpose:

* + Wei\_ReleaseLock

Parameters:

Purpose:

* + Wei\_GetMessages

Parameters:

Purpose:

* + Wei\_MarkRequestError

Parameters:

Purpose:

***Table Name:*** Status

***Purpose:Table:*** Table holds the list of all valid status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int |  | Y | N |
| Name | NVarchar(255) |  |  | N |
| CreateDate | Datetime | Date and time the record was created |  | N |
| CreateOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

***Table Name:*** OFACResponse

***Purpose:Table:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Primary Key** | **Nullable(Y/N)** |
| Id | Int | Unique id for the OFACResponse | Y | N |
| Requested | Int | Id of the request |  | N |
| Responsebody | text | Unmodified OFAC response |  |  |
| CreateDate | Datetime | Date and time the record was created |  | N |
| CreateOper | Varchar(20) | Operator code of the operator who created the record |  | N |
| LastModify | Datetime | Date and time the record was modified |  | Y |
| LastOper | Varchar(20) | Operator code of the operator who last modified the records |  | Y |

***Stored***

Following stored procedures will be created

* + Wei\_AddOFACResponse

Parameters:

Purpose:

* + Wei\_UpdOFACResponse

Parameters:

Purpose:

* + Wei\_DelOFACResponse

Parameters:

Purpose:

* + Wei\_SelectOFACResponse

Parameters:

Purpose:

* + Wei\_SelectOFACResponsebyId

Parameters:

Purpose:

### **Logging and Exception Handling**

WEI™ will use Microsoft Enterprise Library V5.1 for logging. All the trace events will be written to the log file. The path to the log file will be specified in the WEI service configuration file. All fatal errors are written to Windows Event viewer in addition to the log file.

Class File: LogUtil.cs

Purpose: LogUtil.cs is the class file associated with logging

Methods:

logDebug(String message)

logInfo(String message)

log(String message, Exception e)

logError(String message)

logFatal(String message)

log(String message, TraceEventType severity)

### **Security**

Not in this release

## Packaging

A self extracting exe will be created for installing the product.

## Limitations

The tags to be searched are not specific to message type

For XML messages, we will not be able to specify a specific XML tag to search for.