EDISPHERE

EDI Implementation Tasks

EDISPHERE is a suite of very comprehensive any-to-any Electronic Data Interchange (EDI) translation products, comprising of - Translator, Implementor and Collaborator. Our development strategy consisted of using a *whole-product* approach, which placed strong emphasis on its ability to accommodate non-standard and proprietary situations, which are the mainstay of EDI implementation projects.

EDISPHERE's Any-to-Any architecture is designed to meet your data interchange requirements, whether they be for X12, EDIFACT, XML, proprietary messages (fixed length, variable length, CSV); or mapping directly to databases.

EDISPHERE products are horizontally oriented (industry-neutral), designed to accommodate the needs of vertical industries such as Automotive, Banking and Financial, Government, Healthcare, Insurance, Manufacturing, Retail, Transportation/Logistics, etc.

Target Audience: Managerial and Technical

By

Ajay K. Sanghi

October 2005
(Last Updated: October 2007)

www.edisphere.com

EDISPHERE Software is a privately held company, headquartered in Nagpur, India. More Information is available at it's website http://www.edisphere.com

Please Note: This document contains information that is accurate to the author and may contain forward-looking statements with the intent of informing the reader on the vision of EDISPHERE Software. Readers are advised to use their judgment in determining the accuracy and usefulness of EDISPHERE products by requesting an evaluation copy, which can be freely downloaded from EDISPHERE Software's website.

Revision History

Date	Version	Description	Author
09-09-2003	1.0	Initial Version for EDISPHERE 2.0	Ajay K .Sanghi
1-10-2005	1.1	Change format and content for EDISPHERE 3.3	Ajay K .Sanghi
10-10-2005	1.2	Linked white paper to help file	Ganga Sridhar
17-10-2007	1.3	Update for EDISPHERE4.0	Ganga Sridhar

Table Of Contents

Summary

Welcome! At EDISPHERE Software, we have developed very innovative suite of data translation products to “Automate your B2B Operations”.

Businesses communicate messages via post, fax and email. The recipient manually handles the information, which is slow, costly, error-prone and unreliable. The problem is more severe in supply-chain, where based on messages; every party streamlines their operation.

The solution exists in integrating data translation products with your business application, but high cost of implementing Electronic Data Interchange (EDI) technology continues to be a major barrier, particularly when data interchange formats are traditional EDI formats such as X12 and EDIFACT.

EDISPHERE, our suite of any-to-any data translation products makes EDI implementation simple and affordable. It seamlessly integrates with your business application and EDI-enables it to exchange business messages with your customers, vendors and partners; in different formats such as X12, EDIFACT, XML, and Proprietary formats. No programming is necessary for either mapping or application integration.

Unlike other competing products, EDISPHERE has productivity features such as Analyzer, Test Data Generator and Simulator that helps in implementing EDI faster and more robustly. Also, EDISPHERE has features such as Implementation kit, which provides a packaging model for deploying the EDI implementation carried out locally or remotely and also subsequent maintenance/support of it.

EDISPHERE has a high performing multi-threaded translation engine, which can translate about half million 4k size messages every day. Unlike most competing products, EDISPHERE can translate large size EDI messages running into 100s of mega bytes in quick time. EDISPHERE has built-in support for automatically generating and reconciling functional acknowledgement messages. EDISPHERE mappings are backward compatible with new EDISPEHRE releases and also supports reuse thereby safeguarding your years of investment.

EDISPHERE’s very reliable error reporting capabilities makes it highly suited for very demanding EDI implementations.

Our customers in US, Singapore, Hong Kong and India have greatly benefited by improving operational efficiency and reducing transaction cost, besides improving information management, business processes and overall competitiveness of their business.

Our high quality and low cost development in Nagpur provides unbeatable value to our customers. Not only EDISPHERE provides faster return on your investment at lower cost of ownership but you will find us very reliable to adapt to your urgent needs.

Document Organization

This document is organized in a manner to quickly understand EDISPHERE suite and start using it without formal training. It is assumed that EDISPHERE is already installed on your system. The document can be used by organizations that are either evaluating EDISPHERE (pre-sales) or by EDISPHERE customers (post-sales) undertaking EDI implementation.

The document starts by briefly introducing the different data formats supported in EDISPHERE and the possible conversions (mapping) between different data formats in form of a matrix.

The document then describes how the three products - Translator, Implementor and Collaborator; that comprise EDISPHERE suite are used in relation to each other. Subsequently, it describes the tasks associated with implementing EDI using EDISPHERE suite. Each task identifies the EDISPHERE products used in completing that task. Also, each task identifies the role and responsibilities of the parties involved (Customer and Developer) and also any associated dependencies in completing the task.
After having described all the tasks in an EDI implementation, the document then describes a typical interaction between the Customer and the Developer during an EDI implementation cycle. The Developer may be internal or external to the Customer's organization.

This document is linked to the EDISPHERE help files for your ease. They can be downloaded together from the link ***www.edisphere.com/collaterals/ImplementationTasks.zip*** for you to refer to the document offline.

Terminology

- **Electronic Data Interchange (EDI)** - It is used commonly for all data formats including traditional EDI formats like X12, EDIFACT; Proprietary data formats, XML, and also Database formats. X12 and EDIFACT data formats will be referred to as 'traditional' EDI formats wherever distinction is necessary.
- **Layout** – It is the schema or data dictionary used commonly for all EDI formats. A 'layout' is same as '**EDI layout**'. '**Application Layout**' semantically qualifies a layout for interfacing EDISPHERE with the business application. '**Partner Layou**t' semantically qualifies a layout for interfacing EDISPHERE with the trading partner.
- **EDI guide** - A layout when available in form of a document is called an 'EDI guide' or '**Layout specification**'. '**Implementation guide**' or '**Partner guide**' semantically qualifies an EDI guide as layout specification of a 'Partner layout'.
- **Implementation Convention (IC)** – It is a partner layout specification conforming to a published traditional EDI Implementation guide, which is a subset of a published traditional EDI directory.
- **Data Element (or Field)** – It is one element (or field) of information in a segment (or record). 'Data Element' is a more 'traditional EDI' term while 'Field' is a more 'database' term but both convey the same information and is used interchangeably.
- **Segment (or Record)** – 'Segment' is a more 'traditional EDI' term while 'Record' is a more 'database' term but both convey the same information and is used interchangeably. Some use the term 'Record' to mean 'Transaction (or Message)' but in this document, the term 'Record' is used to mean 'Segment' only.
- **Message (or Transaction)** – 'Transaction' is a 'traditional EDI' term in X12 standard to mean a business message while 'Message' is a corresponding 'traditional EDI' term in EDIFACT standard. Both convey the same information and is used interchangeably.

Data Formats Supported In EDISPHERE

The basic data formats supported in EDISPHERE are:

1. **Nested Sequential** – It is the underlying hierarchal data format using which most proprietary file formats (or flat-files) and standard file formats are created. The hierarchal data in the Nested Sequential file format can be of fixed-length, variable-length or CSV (comma separated value). The relationship between the different segments in the Nested Sequential file is 'implicit' (a child segment follows a parent segment and hence there is no need to explicitly link parent-child relationship).

   SAP IDOCs, VDA standards (used mainly in Germany by the Automobile Industry) are created using Nested Sequential (fixed-length) file format (note – SAP IDOCs are also available in XML data formats). TRADACOM standards (used mainly in UK by the Transportation industry) are created using Nested Sequential (variable-length) file format. CARGO-IMP messages, which contains a mix of fixed-length and variable-length data can also be created using Nested Sequential file format.

2. **X12** – It is the standard data format used widely popular in US using which various vertical industry associations, large corporations and US government organizations have published their EDI guides. X12 standard messages published in Standard Exchange Format (SEF) can be directly read by EDISPHERE obviating manual creation of these messages.

3. **EDIFACT** - is the standard data format used internationally using which various vertical industry associations, large corporations and government organizations all around the world have published their EDI guides. ODETTE standard messages, which are popular in Europe are created using EDIFACT file format. EDIFACT standard messages published in Standard Exchange Format (SEF) can be directly read by EDISPHERE obviating manual creation of these messages.

4. **XML** – It is the modern data format using which various vertical industry associations, large corporations and government organizations all around the world have published their EDI guides in form of XML DTDs and XML Schemas. If the XML layouts are available in XML DTDs and XML Schemas then they can be directly read by EDISPHERE, else the XML layouts can be manually created with ease. For example, XML DTDs and XML Schemas published for XML data interchange standards such as

Rosettanet, Chem eStandards, xCBL, cXML, UBL, SAP IDOCs, etc. can be read directly by EDISPHERE, obviating manual creation.

5. **MFF** – It is an EDISPHERE specific abbreviation for Multiple Flat File (MFF) format. Like Nested Sequential file format, the hierarchal data in the MFF format can be of fixed-length, variable-length or CSV (comma separated value) and unlike Nested Sequential file format, where the data is confined to one file and the segments are 'implicitly' related; the data in MFF format can be dispersed across multiple files and the segments are 'explicitly' related through primary-foreign key relationships. For example, the parent segments can be in one file whereas the corresponding child segments can be in separate files, which are linked through primary-foreign key relationships.

6. **Level 1 Flat File** – It is a combination of Nested Sequential and MFF format files. Like Nested Sequential file format, it is confined to one file in fixed-length, variable-length or CSV (comma separated value) format but instead of 'implicit' linking, different segments are 'explicitly' linked through primary-foreign key relationships like in MFF format files. For example, all parent segments can be present at the beginning of the file followed by all child segments in the same file linked through primary-foreign key relationships.

7. **Database** – It is a data format supported in EDISPHERE using ODBC and ADO technologies. Many popular databases such as MS Access, MS SQL Server, Oracle, MySQL, DB2, etc. are supported, whose schemas can be read directly by EDISPHERE obviating manual creation. Each table in the database corresponds to a segment in EDISPHERE layout. Excel files are also supported similarly, wherein each Excel sheet corresponds to a segment in the corresponding EDISPHERE layout.

The following table shows "any-to-any" mapping capabilities of EDISPHERE:

Output Format-> / Input Format	Nested Sequential	X12	EDIFACT	XML	MFF	Level1 Flat File	Database
Nested Sequential	√	√	√	√	√	√	√
X12	√	√	√	√	√	√	√
EDIFACT	√	√	√	√	√	√	√
XML	√	√	√	√	√	√	√
MFF	√	√	√	√	√	√	√
Level 1 Flat File	√	√	√	√	√	√	√
Database	√	√	√	√	√	√	√

EDISPHERE has the capability to map envelope information, message information and trading partner agreement information to the output message and envelope.

	Message (Destination)	Envelope (Destination)
Message (Source)	√	√
Envelope (Source)	√	√
Trading Partner Interchange Agreement (TPIA)	√	√

Note:

Trading Partner Interchange Agreement (TPIA) is an EDISPHERE proprietary format for mapping Trading Partner Interchange Agreements (TIPA) to output data formats. For example, the outbound XML to X12 or XML to EDIFACT translation may require mapping TPIA values such as Interchange password, Interchange sender ID, Interchange receiver ID, etc, which do not form part of the XML business message are mapped to X12/EDIFACT envelope segments ISA/UNB and GS/UNG to meet the mandatory X12/EDIFACT standard requirements. **These maps (TPIA -> X12/EDIFACT envelope) are predefined and the user only needs to use them in the trading partner agreement wherever necessary**.

Using EDISPHERE Suite - Translator, Implementor and Collaborator

To undertake an EDI implementation, the EDISPHERE products are used as follows:

a) **Implementor:** Create Layouts and Maps

Create the application layout (normally – proprietary flat-file, XML or database layout) so that Translator can recognize and exchange messages with the business application and also validate its data.

Create the partner layout (normally - X12, EDIFACT or XML layout) as agreed upon with the trading partner so that Translator can recognize and exchange messages with the trading partner and also validate the data contained therein.

Create maps correlating the data in the application layout and partner layout, so that Translator can perform the appropriate conversion between the application and partner message formats.

Implementor is packed with smart and innovative features, which significantly simplifies your EDI implementation effort. At the time of layout creation, syntax validation rules are automatically built-into the layout. Test Data Generator will rapidly generate sample data for testing. Analyzer will validate your layouts while Simulator will validate your maps.

b) **Collaborator:** Create trading partner agreement

Create partner profiles and agreements, so that Translator can exchange messages with trading partners and also to locate the appropriate layouts and map file for conversion.

The Implementation kit is a very innovative feature in Collaborator, used for packaging your entire EDI implementation for local or remote deployment from development to production, and also to support the implementation in production.

c) **Translator:** Perform translation

Configure various servers like scheduler, sender(s), receiver(s), listener, functional acknowledgement, etc. to send and receive messages between your business application and your trading partners in an unattended mode. Create rules for identifying the layout to be associated for analyzing the file based on initial text in the interchange or

file naming convention. Translator uses the files prepared by Implementor (layouts, maps) and Collaborator (trading partner agreement) to perform the translation.

Translator's multithreaded architecture significantly boosts its performance. Its Alert and Notification feature warns you of translation failures via email.

Steps For Creating EDI Implementation

Step 1 - Provide Information for EDI Implementation

a) Provide layout Information such as EDI guides, XML schemas, XML DTDs, etc. for creating application and partner layouts. If Application layout is a Database then provide SQL scripts for creating the database.

b) Provide **application integration** information for exchanging messages with your business application (folder names, file naming conventions, etc). Many applications (especially on mainframes) exchange application messages using ftp protocol. Specify ftp user name, password, folder names, etc.

c) Provide mapping information for mapping application layout fields with partner layout fields. Provide cross-reference (lookup) information for mapping partner codes differently with your application codes (product ids, vendor ids, currency codes, etc).

d) Provide envelope Information, especially if traditional EDI messages are involved, such as ISA or UNB IDs of both yourself and your various partners.

e) Provide **Partner Integration Information** (or communication information) for exchanging messages with your partner (ftp user name, password, file naming convention, etc). If you plan to use a third party tool for communication (say for AS2 communication) then specify the folder names for exchanging partner messages.

f) Provide information related to handling of errors. For example, do you require an email alert be sent to some specific email address whenever an error occurs?

g) Provide lots of sample data (if possible) for both input and output messages for robustly testing your EDI implementation. Clearly specify the acceptance test scenarios, which should include different error scenarios and acknowledgement responses.

Step 2 - Create Application Layout

Using Implementor,

1. Use Implementor | File | New or File | Open feature (as the case may be) to create Application Layout.

   Application data can be of any data format. It is normally XML, Flat-File (Nested Sequential – Fixed length, Variable length or CSV) or Database (MS Access, MS SQL Server, Oracle).

   EDISPHERE supports X12 and EDIFACT formats also as Application formats, which may be required by Value Added Network (VAN) customers.

   Based on your application data format create your application layout. Follow the link **Create Layout** *in Implementor Help* to find out how to create your Application layout.

2. Generate Test Data For Application Layout.

   Use Implementor | Tools | Test Data Generator feature to generate sample data.

   For inbound scenarios, generate test data using application layout and feed it to your application for further processing. You can generate random data or provide specific data in the Test Data Generator. You can also generate multiple instances of your line items. Take appropriate action based on errors reported by Test Data Generator and your application.

   Even for your outbound scenarios, you may want to generate test data and take appropriate actions if there are any errors.

   Follow the link **Generate Test Data** *in Implementor Help* to find out how to generate test data for your application layout..

3. Analyze sample input data with the application layout

   Use Tools | Analyzer to analyze your input data.

   If you already have Application input data, then analyze input data with the Application layout. Take appropriate action based on errors reported by Analyzer. If you do not have input data then use data generated through Test Data Generator

   For outbound scenarios, your application will generate the data for input to the Analyzer. Even for your inbound scenarios, if you have data then analyze it to check if there are any errors and take appropriate action.

   Note – In case your Application layout is Database then it is better to first generate test data before using Analyzer. Using Analyzer and Test Data Generator ensures that your data format can be appropriately read and written by EDISPHERE besides informing you About any data validation errors.

   Follow the link for **Analyze** *in Implementor Help* for validating data corresponding to your Application layout.

Step 3 - Create Partner Layout

The process for creating Partner layout is same as creating Application layout.

Using Implementor,

1. Use Implementor | File | New or File | Open (as the case may be) to create Partner Layout.

   Partner data can be of any data format. It is normally XML, Flat-File (Nested Sequential – Fixed length, Variable length, or CSV), X12 or EDIFACT.

   EDISPHERE supports Database formats also as Partner format, which may be required for your internal multi-application integration scenarios.

   Based on your partner data format create your partner layout. Follow the link for **Create Layout** *in Implementor Help* to know how to create your Partner layout.

2. Generate Test Data For Partner Layout.

   Use Implementor | Tools | Test Data Generator to generate sample data.

   For inbound scenarios, generate test data using partner layout and feed it to Analyzer. You can generate random data or provide specific data in the Test Data Generator. You can also generate multiple instances of your line items. Take appropriate action based on errors reported by Test Data Generator and Analyzer.

   Even for your outbound scenarios, you may want to generate test data and take appropriate actions if there are any errors.

   Follow the link **Generate Test Data** *in Implementor Help* to know how to generate data corresponding to your Partner layout.

3. Analyze data with the Partner layout

   Use Tools | Analyzer to analyze your input data.

   If you already have Partner input data, then analyze input data with the Partner layout. Take appropriate action based on errors reported by Analyzer. If you do not have input data then use data generated through Test Data Generator.

   For inbound scenarios, your partner will provide input data for testing. Request sample data files from partner even for your outbound scenarios.

   Follow the link **Analyze** *in Implementor Help* for validating data corresponding to your Partner layout.

Step 4 - Create Mappings between Application Layout and Partner Layout

Using Implementor,

1. Use Implementor | File | New Map file option to create a new map file. Specify Application layout as the first layout (recommend practice) and then specify your partner layout as the second layout.

2. It's a good idea at this point to print Maps! The printout will contain only application layout fields (in layout 1 position) with nothing filled in the layout 2 position.

3. Manually fill this map printout with the corresponding layout 2 fields so that you can quickly map using Implementor user interface, by referring to this document.

   For both your Inbound and Outbound mappings create your maps. Follow the link **Create Maps** *in Implementor Help* to know how to create maps.

4. The drag and drop mapping user interface provides several advance mapping capabilities including creation of cross-reference (lookup) tables.

5. Save the map file with an appropriate (meaningful) name and description. When opening the map file again, a meaningful description will help you identify your map file quickly.

Step 5 - Create Trading Partner Agreement

Using Collaborator,

Use Collaborator| File | Create Partner to create partners. Each partner can be a Sender or Receiver of the EDI message based on which appropriate trading partner agreements are created. For traditional EDI standards, X12 and EDIFACT, the partner ID will correspond to the Sender/Receiver ID in the ISA segment (X12 standard) and UNG segment (EDIFACT standard). Follow the link **Create New Partner** *in* Collaborator *Help.*

1. Each Partner node represents a Sender. For outbound documents, you are the Sender and your partner is the Receiver. For your inbound documents, your partner is the Sender while you are the Receiver.
2. Select the appropriate Sender node and use Collaborator | File | Create Agreement to create Trading Partner Agreement.
3. Select appropriate Source format -> Destination format Translation Type.
   The choices are:
   a. EDI (Envelope)->EDI (Envelope)
   b. EDI (Envelope)->Proprietary (No Envelope)
   c. Proprietary (No Envelope) -> EDI (Envelope)
   d. Proprietary (No Envelope) -> Proprietary (No Envelope)

   If a data format have envelopes associated with it like in the case of X12 and EDIFACT standards then select EDI (Envelope) else select Proprietary (No Envelope). All XML, MFF and Database formats are examples of Proprietary (No Envelope) cases. EDISPHERE supports creation of Proprietary envelopes for Nested Sequential formats, however, most usage is that of Proprietary (No Envelope) cases.
4. Based on the selection of Translation Type, either source standard or destination standard is to be selected.
   The choices are:
   a. X12 standard
   b. EDIFACT standard
   c. Other EDI (Envelope)
   d. Proprietary (No Envelope)

   The user is asked to make this selection in order to provide friendly labels for accepting input from the user, particularly when using X12 or EDIFACT standards. If X12 is selected then user sees labels corresponding to ISA and GS envelope segments while if the selection is EDIFACT then user sees labels corresponding to UNB and UNG envelope segments.
5. The Receiver partner is selected from the list of already created Partners in the field “Interchange Receiver ID” (envelope information section).
6. “Workflow” is the most significant information in the trading partner agreement. The basic workflow consists of the following:
   a. Analyze source interchange (always used)
   b. Map source message to destination message (always used)
   c. Map agreement to destination envelope (always used in outbound scenarios when the destination format is X12 or EDIFACT standard. User does not have to create maps for Agreement -> X12/EDIFACT envelopes as these maps are already provided)
   d. Generate destination interchange (always used)

   EDISPHERE workflow also supports mapping cases such as “Map source message to destination envelope”, “Map agreement to destination message”, “Map source envelope to destination envelope”, and “Map source envelope to destination message”, all of which are rarely used.

7. While specifying map files in the workflow, provide mapping direction as Lay 1 -> Lay 2 for outbound maps and Lay 2 -> Lay 1 for inbound maps (assuming Application layout was selected as layout 1 and Partner layout as layout 2 as recommended earlier; else the directions will have to be reversed).

   Follow the link **Create New Partner Agreement** *in* Collaborator *Help*

.

Note – A diagonal bar is shown on the Receiver node of the agreement if the agreement is incomplete (i.e. some mandatory information is missing).

Step 6 – Test - Simulate To Verify EDI Mapping

Using Implementor,

1. Use Implementor | File | Open Map File to open the map file that has been created. Use Implementor | Tools | Simulate to test your mapping. Follow the link **Perform Simulation** *in Implementor Help* to know how to simulate.

2. Verify your mapping with the Simulator output, which shows the following:
    a. All information that has been analyzed from the source file including any validation errors.
    b. All information that has been mapped corresponding to each source element including any operation such as padding, truncation, string operation (like concatenation of data), or mathematical operation (like multiplying the result of 2 elements in the source and mapping the result to the output element) performed before mapping to the output element.
    c. All information that has been generated and written to the output format.

    Mapping errors can be quickly identified and corrected with Simulator output.
    Note - Ensure that you have saved the map file before simulating it again; else the changes to the maps will not trickle to the Simulation process.

Step 7 – Create Source File Identification (SFID) Rules

Using Collaborator,

1. Use Collaborator | Tools | Source File Identification to create rules for source file identification, which is necessary for the purpose of:
    a. Identifying the layout using which the source file will be read and validated. These SFID rules are mandatory for all data formats.
    b. Identifying appropriate trading partner agreement (EDI mappings). These SFID rules are required only if the necessary information to identify a trading partner agreement is not available in the data file.

    In case of X12 and EDIFACT data formats, the starting three characters (segment IDs) of the data interchange file unambiguously determine these standards (ISA segment in case of X12, UNA or UNB segment in case of EDIFACT). the same is not true for most proprietary files.

    Once the layout is determined, it is then necessary to determine appropriate trading partner agreements for different messages in the same interchange file. The minimum information required for identifying appropriate trading partner agreement is “Sender”, “Receiver” and “Message Type”.

    In case of X12 and EDIFACT standard, the “Sender” and “Receiver” information is specified in the envelope segments of the data interchange file (ISA segment in case of X12, UNB segment in case of EDIFACT) and “Message Type” is specified in the first segment of the message (ST segment in case of X12, UNH segment in case of EDIFACT).

    The source file identification rules are preset for X12 and EDIFACT data formats. A source file can be identified using one or more of the following criteria:
    a. File naming convention – For example, SFID rules may be created to identify files in various folders with file name pattern “ABC-XYZ-PO*.xml” as XML files corresponding to PO-XML layout with trading partner agreement corresponding to Receiver ABC, Sender XYZ and message PO.
    b. Directory naming convention – For example, SFID rules may be created to identify all files in the folder matching the pattern “ABC\PO” correspond to PO-XML layout with trading partner agreement corresponding to Receiver ABC, Sender XYZ and message PO. Note – in this case the sender XYZ is assumed.
    c. Segment ID convention – For example, if the first three characters of a file are “ISA” then the file is to be read using X12 envelope layout. Similarly, if the first three characters of a file are “UNA” or “UNB” then the file is to be read using EDIFACT envelope layout.
    d. Segment version convention – For example, the X12 files identified by reading first 3 characters as “ISA”; if the version specified in the file is 00401 then use file layout corresponding to X12 envelope version 4010 else use file layout corresponding to X12 envelope version 3050.

    Source file identification rules are preset for X12 and EDIFACT file formats. To specify rules for other file formats, follow the link **Source File Identification** *in Translator Help.*

    Note – Source File Identification rules can also be specified using Collaborator at the time of creating trading partner agreement in Analyze work of the workflow.

    As a guideline, rules that identify an agreement should be specified while creating trading partner agreement, while rules that identify only layouts (rules common to multiple agreements) should be specified using Translator.

Step 8 – Test - Translate To Verify EDI Implementation

Using Translator,

1. Manual Translations - Use Translator | File | Import to manually translate the source file. If no entry is seen in the Inbox folder for this translator then something went wrong while creating the file identification rule. Correct it and translate again. After the translation is successful, check the corresponding output file.
2. Automated Translations - for testing translations in automated mode requires setting up the following:
    a. Use Translator | Tools | System Options | Receiver to setup appropriate Receiver directory from where the source files will be picked up automatically by the Translator.
    b. Use Translator | Tools | System Options | Server to setup appropriate Servers - Receiver, Scheduler and Listener. Subsequently, start the servers using Translator | Tools | Start Servers

    After sometime, the input source files will be automatically picked up for translation and corresponding entries will visible in the Inbox and Outbox folders of the Translator's User Interface.

    Follow the link **Setup Translator for Automated 24x7 Translations** in Appendix to setup translator for testing in automated mode.

Step 9 – Package EDI Implementation

Using Collaborator,

Use Collaborator | Tools | Create Implementation Kit to package the EDI implementation for the desired trading partner agreements, which will package all dependencies related to the selected trading partner agreements (layouts, maps, source file identification rules, workflow, cross-reference tables, etc) in the “Impkit” folder. The files in the “Impkit” folder can be zipped and sent as implementation kit. Follow the link **Create Implementation Kit** *in* Collaborator *Help* to know how to create Implementation kits.

Implementation kits are created on the Development PC for deployment on Test PC and Production PC. It is recommended that all changes related to implementation kits are always carried out on the Development PC and deployed using the Implementation kit model rather than making changes directly on the Production PC.

Step 10 – Deploy EDI Implementation

Using Translator,

Use Translator | Tools | Import Implementation Kit to deploy the EDI implementation. Both new EDI implementations and upgrades to existing EDI implementations can be deployed using Implementation Kits. Follow the link **Import Implementation Kit** *in Translator Help* for deploying Implementation kits.

While deploying Implementation kits, Translator warns the user if older files in the kit are overwriting any new files. Unless you are sure about overwriting the newer files always say “No” or “No To All”. Older files can be safely overwritten by new files but not vice versa.

Step 11 – Field Test – Translate Test Messages

It is normal practices in EDI space to first field test the messages prior to going Live for production use. Different companies have different strategies for field-testing. One or more of some of the following strategies are common:

a. Use Production/Test field in X12/EDIFACT messages to distinguish between Test and Production use.
b. Use different Sender and Receiver IDs in X12/EDIFACT messages to distinguish between Test and Production use.
c. Use a set of sample EDI messages (both inbound and outbound), which tests capabilities of responding to EDI messages besides translation and acknowledgement capabilities.
d. Exchange production use messages directly and rely on parallel existing business process along with functional acknowledgement messages to field test the EDI messages before fully obsolescing the existing manual process.

EDISPHERE has features to support all such scenarios.

Step 12 – Go Live – Translate Production Messages

The Translator on the Production PC should be operated in 24x7 automated translation mode; in either foreground or in background as “service”. For assistance, follow the link **Setup Translator for Automated 24x7 Translations.**

The Translator should be appropriately setup for reporting errors using Alert and Notification feature (see Translator | Tools | Alerts and Notifications), and periodically observing manually for any errors by looking at Translator logs (in particular observe Program Exception and Resource Exception logs for anything unusual). See **EDISPHERE best practices for monitoring live EDI Implementations**.

On the Production PC, only Translator should be used and all EDI implementations should be deployed and upgraded using the Implementation Kit model (recommended practice) rather than using Implementor and Collaborator directly on the Production PC. See **EDISPHERE best practices for maintaining EDI implementations**.

Conclusion

Implementing EDI using EDISPHERE is simple and consists of 5 basic steps:

1. Use Implementor to model your application layout for application-EDI interface.
2. Use Implementor to model your partner layout for partner-EDI interface.
3. Use Implementor to map your application layout with your partner layout.
4. Use Collaborator to specify partner information and communication parameters.
5. Use Translator to start exchanging messages with your trading partner.

In between the above steps, use Implementor's productivity features like Analyzer, Test Data Generator and Simulator to test your layouts and maps. After completing acceptance testing, package your EDI implementation using Collaborator 's Implementation Kit feature, for deployment and support. Please send your feedback on the white paper directly to the author as *ajayksanghi@edisphere.com* It will be highly appreciated.

Appendix A : Application Integration Interface

EDISPHERE runs on Windows but it can seamlessly integrate with any business application running on any platform using one or more of the following options:

a) **Common Directory (or Folder)** – EDISPHERE::Translator and your Application can exchange messages through common folders using a data format most suited to your Application. The folders and filenames may follow a naming convention most suited for your application based on partner names, business messages, direction of the message, etc. One or more folders may be shared for application-integration and also the folders may be hierarchal.

   EDISPHERE::Translator will automatically pick up the outbound files generated by your business application from the common folder. Similarly, Translator will deposit the inbound files received from your trading partner following their translation (mapping), in a common folder.

   The common outbound folders can be specified as Listener directories in Translator for automatic pick up. Follow the link **Listener Settings** in *Translator Help*

   Different business messages intended for different partners may be present in "one" common folder. In such cases, appropriate application layout and mappings are identified based on file naming convention.

   Source File Identification (SFID) rules based on file name pattern and/or common folder name pattern can be specified in Translator. Follow the link **Source File Identification** in *Translator Help.*

   Note - If your application is residing on a different platform such as Mainframe, it is common to use FTP for exchanging files with the common folders.

b) **Common Database** – EDISPHERE has the capability to map directly to/from Databases. Hence databases may be used for application integration instead of files. Database can be shared with EDISPHERE by creating Data Source Name (DSN) for the common database. Follow the link **Application Integration For Database layouts** in *Implementor Help.*

c) **EDISPHERE API** – EDISPHERE exposes APIs in C, C++, Java, JMS, COM, and also through web services for triggering translation from within your own business application. The APIs are used for identifying appropriate source files or database keys for outbound operations and to query Translator for any inbound files. Several other EDISPHERE capabilities are exposed through the APIs. Follow the appropriate link in **API help**.

Note – Sample programs demonstrating use of EDISPHERE APIs are also available.

Appendix B : Create Partner Integration Interface

EDISPHERE has the built-in capability to exchanges files with your partners using FTP and Email. For other communication options are desired either because of the corporate policy to always use corporate communication server or a different communication program (say for EDINT-AS2 communication), EDISPHERE can seamlessly exchange files with external communication program through common folders (or FTP).

a) Outbound Partner integration interfaces are specified at the time of creating trading partner agreement in Collaborator. Follow the link for Transmit Destination Interchanges in Collaborator help. Using Collaborator configure the Trading Partner Agreement's Transmit Destination Interchanges work detail to send the translated message in the form of **E-Mail , FTP or to a common directory**. Follow the appropriate links to **WorkFlow** *in* Collaborator *Help*

b) Inbound partner integration interfaces are specified using Receivers in Translator. Follow the link **Receiver Settings** *in Translator Help*

Appendix C : Setup Translator for Automated 24x7 Translations

Using Translator,

(i) Select the **File | Import** option. Specify the input data file as the input file for translation.

(ii) Select **Tools | System Settings | Receiver** tab and setup appropriate Receiver directory where flat-files will reside. Specify the required folder as the receiver directory.

(iii) Click on **Tools | System Settings | Server tab**. Check the "Generic Timer ",“Scheduler” ,"Directory Receiver" and “Listener” checkbox. TRANSLATOR will periodically (automatically) translate all the interchange files it has received for translation and send them to the appropriate destination as configured in the Partner agreement.

(iv) On Completion of the outbound translations you can view message entries for the analyzed files in the Translator’s Inbox and the corresponding entry for the generated files in the Translator’s Outbox

Appendix D : A Typical Interaction Between The Customer And Developer

a) Customer sends (emails) the Specification (all information pertaining to creation of Implementation) to Developer.
b) Developer verifies the information received and asks questions to Customer if any clarification is necessary by emailing the questions to Customer.
c) Customer may have to in turn ask questions to Partners to respond to Developer.
d) Developer starts the Implementation. Developer may ask more questions to Customer while undertaking Implementation and Customer would like wise respond.
e) Developer sends Implementation kit to Customer. Developer ensures that the Implementation kit meets the Specification. Developer sends both input and output file of the testing carried out by the Developer to meet the Specification.
f) Customer verifies that the Implementation kit sent by Developer meets the Specification. If Customer finds the Implementation kit does not meet the Specification, the Customer notifies the same to the Developer.
g) Developer resolves the issues and resends the Implementation kit to the Customer.
h) If Customer does not report any issues to the Developer for two (2) weeks of delivering the Implementation kit, the Implementation is deemed accepted by the Customer.
i) After acceptance of the Implementation, at some later date, Customer requests a change in the Specification or encounters an error in the Implementation. The Customer notifies the same to the Developer by sending Implementation kit of the Implementation in which changes are to be made, along with data files.
j) Developer makes appropriate changes in the Implementation and sends back the Implementation kit to the Customer.

About the Author

Ajay K Sanghi is the Founder, CEO/CTO of EDISPHERE Software since it's inception in 1995. He is a hands-on executive, who continues to passionately contribute in design and coding of EDISPHERE. Ajay is also very passionate about the Company's CSR program.

Ajay has a Bachelor's degree in Electronics Engineering from India and Master's degree in Computer Engineering from USA. Prior to starting EDISPHERE Software, he has worked with the R&D team of leading Telecommunication companies in USA.

Ajay is a vegetarian, loves to jog and practices yoga and meditation. His favorite place is his hometown, Nagpur, right in the center of India, where he was born and grew up with lots of friends. He may be reached at ajay.sanghi@edisphere.com

About EDISPHERE Software

At EDISPHERE Software, we develop Innovative EDI products for automating supply-chain.

Large companies and governments are using EDI technology to improve their operating efficiency and reduce transaction costs. But the high cost of implementing EDI technology continues to be a major barrier.

EDISPHERE is a comprehensive suite of innovative Electronic Data Interchange (EDI) products for seamlessly integrating internal business applications with external partners, which helps automate supply-chain faster, robustly and more cost-effectively. Based out of India; it's high-quality R&D and low-cost support provides unbeatable value to it's customers in US, UK, Singapore, Hong Kong and South Africa. Additional information about EDISPHERE can be obtained at http://www.edisphere.com

Contact:

EDISPHERE Software Private Limited
215 Congress Nagar, Nagpur 440012, India

Tel (India): +91 712 246 3314
Tel (USA): +1 408 (649)-5635 (Uses VOIP – call answered by Nagpur office)

Fax: +91 712 246 3315

Email: info@edisphere.com