Etasoft
Building blocks of integration

Guide on EDI automation and use of VAN services

Main website http://www.etasoft.com
Extreme Processing website http://www.xtranslator.com

Basic Requirements

This document describes some setups and scripts for EDI processing using Extreme Processing product.

Once script is done you do not have to recreate it again simply save it into the file with extension *.run. You can run script files using other utilities that come in the package (read User's Manual about other utility programs).

Software Requirements

You will need to download and install Extreme Processing from the website http://www.xtranslator.com . Extreme Processing comes with built-in tasks for Extreme Translator, Extreme Validator and many other processing tasks.

Once it is installed, start Script Editor tool.

Mailbox

EDI setups usually are based on "Mailbox" concept. Mailbox is nothing more than a directory on local drive, network server or FTP folder somewhere on the Internet. Mailbox is dedicated to handle one type of message. Example: mailbox for EDI X12 810 Invoices. Mailbox only handles incoming or outgoing files. So in our case let's say it is "mailbox for incoming EDI X12 810 Invoices on local drive".

Since most EDI translators work on files matching certain pattern, once you run translation, it will know what translation map to use against this specific directory ("mailbox") and what pattern to look for. Every message type gets its own mailbox. So if you need to receive EDI X12 810 Invoices and send out EDI X12 997 Acknowledgements you would have one mailbox for 810s and one for 997s.

Now imagine you have all your files in one directory and you have no idea about this "mailbox" concept. Every time you look at the directory contents you see long list of files and you have no way of knowing where you have 810s and where are 997s. Only way to distinguish them would be to have specific file naming convention or manually open each and every one of those files using text editor. That's not so bad right? But imagine that you have many different trading partners sending you many message types. You can not force everyone to adopt your file naming convention. Next thing you know your processing directory is clogged with all kinds of files. Now real nightmare begins – once some translations or validations fail you need to find what went wrong by going thru thousands of files in one big giant directory. Even if you have perfect error reporting facilities, things just tend to become hard to track if all files are in one big mesh.

While not that common some trading partners may send you files with the same names. Chances are slim but you would not want one trading partner overwriting files from another trading partner. Since all your files come into one directory it is possible.

If you setup mailboxes translator does not need to waist time going thru all the files, read they ISA/IEA envelopes one by one trying to find out what message type is in the file, who is sender and receiver and what translation map should handle it.

There are many other reasons why separating files by message type and direction (incoming vs. outgoing) is a good idea. Mailbox concept simply makes EDI setup much more cleaner. It is easy to setup and does not require any special training or software to manage.

Gatekeeper

Many manual EDI setups where data is processed by personnel do not have EDI validation and in most cases do not even have basic sanity checks. Assumption is if data is processed and posted into the system – it is good. Some reliance is placed on human operator to spot invalid files, example: if file has size of zero bytes – it is invalid. While this works in real implementations for manual EDI processing, "no validation" makes automated systems brittle and unreliable.
Validation or at least set of sanity checks is important part of any automated system. You can design the system so only validated files pass validation step and all others are reported and posted for review or automatically rejected and some acknowledgement is sent back to the sender. Validation task is a gatekeeper. Once files or data passes through that point in the system it is safe.
"Gatekeeper" concept is important because it also splits automated system into two parts: external and internal. External part works with all the files submitted for processing. Internal part only works with "clean" files and assumes that data is valid.

Plan Your Setup

The way you setup automated processing can save you time, resources and money. If setup improperly automated processing can also be constant source of problems, and then it can cost time and resources.
Initial stage of setup is most important because fundamental decisions are usually made at that stage. Decisions that influence system setup can easily change during this time.

Concepts are important. Like "mailbox" concept explained in chapter above. Systems built on strong concepts are more reliable.

It is a good idea to draw setup on a piece of paper or use flow chart software to put initial blocks and have some design diagram to base your thinking process. The design may evolve and change but at least some basic documentation trail will keep you on track.
Do not start developing the system until you have clear picture of how all the parts of it should work together.

There are some main points to consider when developing automated processing system:

1. File system. How will files flow through the system? You need to setup all the steps in the lifetime of the file such as where you are going to get files from and/or where you will submit them to?
2. Events. What events trigger processing? Is it time schedule based processing that wakes up once at 2 AM and checks for the files to process or is it file trigger based processing when new incoming file causes system to wake up?
3. Exceptions. What happens when errors occur during processing? Keep in mind that usually there are at least few different kinds of errors: data validation errors, critical system errors, etc.
4. Logs and reports. What logs and reports will processing produce?

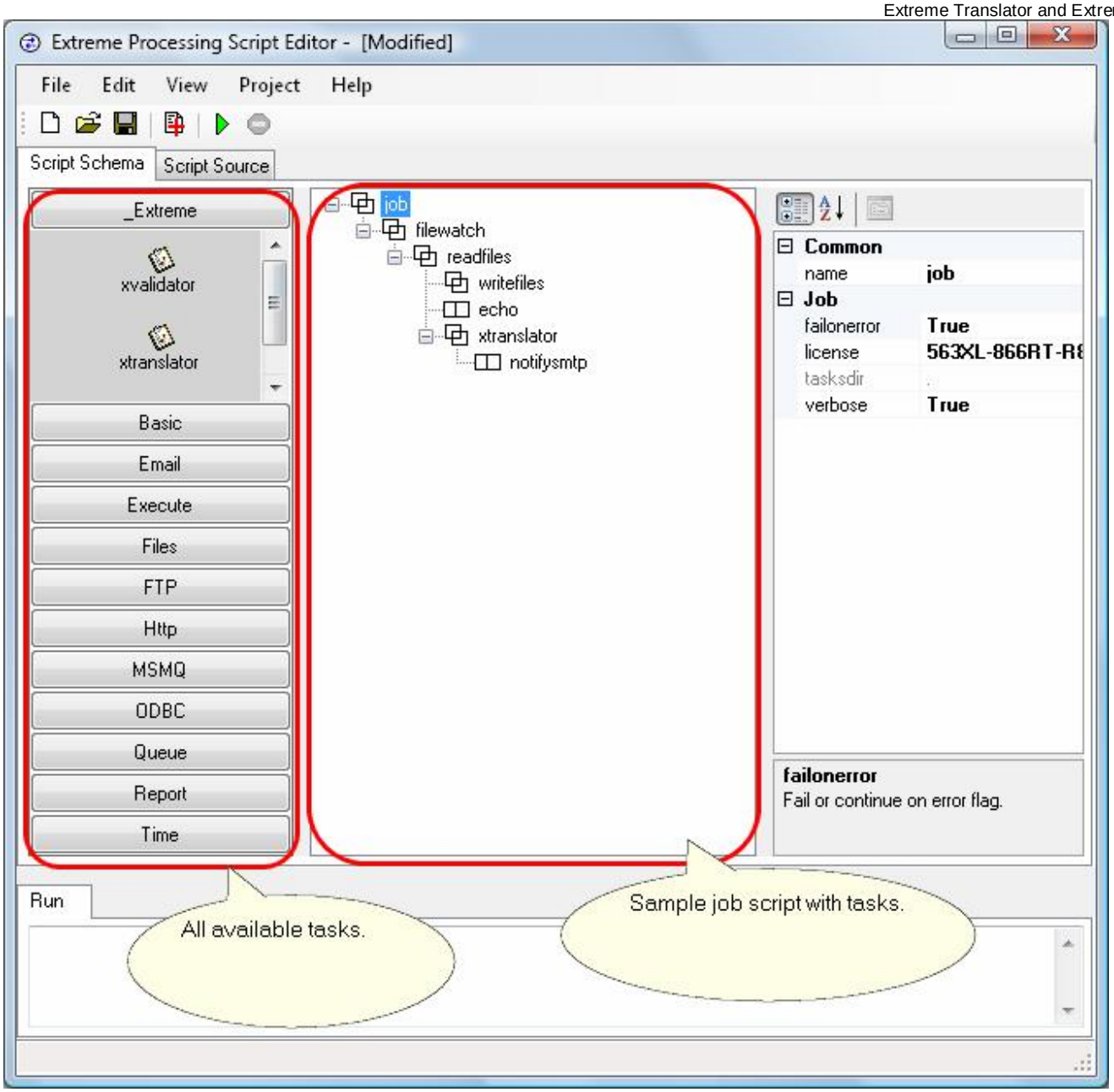
Meet Extreme Processing

How does Extreme Processing product fit in processing system automation?
Extreme Processing product is built around concepts of job and task. Job is script that runs your automated system. Task is individual unit of work that job executes on each processing step. While technically you can have many jobs/scripts to run to perform required processing most systems only need one script.
Extreme Processing does not force you to design your processing and file system in some special predefined way in order to automate your system. That is, you can setup your job with tasks anyway you want. However we do recommend you to use "mailbox" and "gatekeeper" concepts outlined in this document when you are setting up your job script.
Because Extreme Processing does not enforce specific design you can match your specific requirements to system process. Example: if you need to backup all incoming files, simply setup separate file directory and have Extreme Processing <movefiles> task to copy all incoming files into this archive directory.

Script Editor

Extreme Processing comes with script editor. You can setup jobs and add tasks to them using this graphical tool. Number of tasks supplied with the product allows various processing scenarios. There are tasks that operate on files, run data validation, translation, wait for system file or time events, write processing logs, send email notifications, and much more. All available tasks are grouped based on they category.

If you are using Extreme Translator most of your job scripts will contain <xtranslator> task.

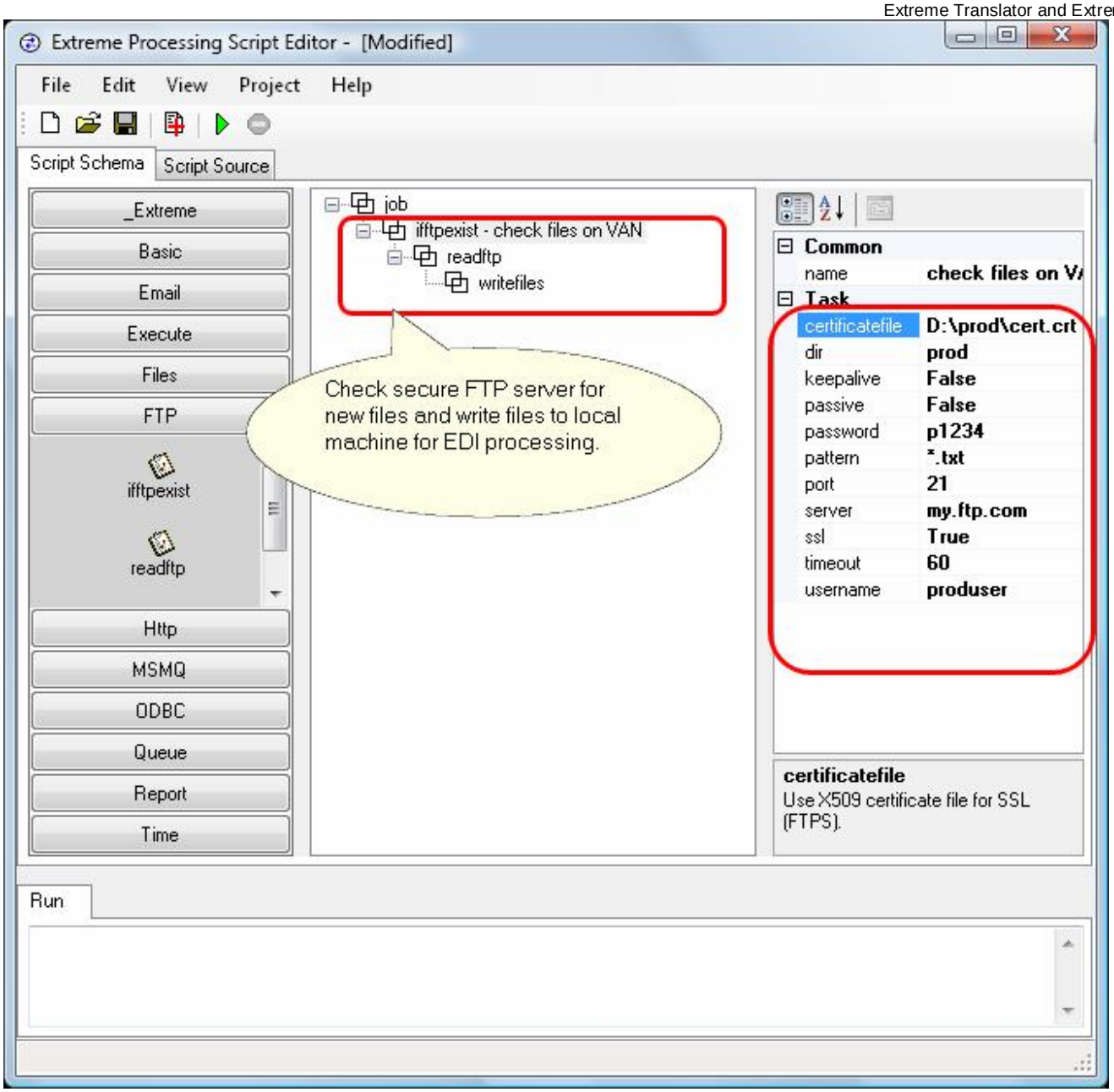


This sample job script will watch file system for incoming files, once file comes in it will read and move files to processing directory based on file name pattern using <readfiles> and <writefiles> tasks (only *.txt text files will be processed), then it will execute Extreme Translator <xtranslator> task and send email notification if translation is successful.

EDI VAN

You might have to exchange data with trading partners that use VANs (Value Added Network). In old pre-Internet era days VANs were almost only way to send and receive EDI data. Other way was to copy EDI data on to the floppy disk and mail it… seriously. The way to connect to the VAN was to dial-in with the dialup connection to the dedicated server, establish connection and get access to the EDI mailbox.

These days VANs are replaced by FTP, secure FTP or AS2 servers or other Internet based technologies. Some companies still call those services “VAN” to indicate that it is not just simple secure FTP server that you can connect to using virtual private network connection over the Internet, but it is FTP server specifically dedicated to EDI processing.
This terminology is misleading because customers not familiar with history of EDI processing systems get confused thinking that “VAN” is some magic technology that you need special tools to tackle.
However there might be the case when you actually need to connect to real VAN on some old mainframe and use old communication protocol (not FTP or AS2) to get your data from EDI mailbox.



This sample script connects to "VAN" using secure FTP over SSL connection via Internet and pulls files from server into local machine for EDI processing.

EDI Mailbox Structure

EDI mailbox is an abstract concept representing temporary storage for EDI files. It is point of EDI file exchange between two trading partners. Files move in and out of mailbox weekly, daily or hourly. In most cases EDI mailbox is nothing more than FTP server directory, AS2 or web service. Those are usually accessible via Internet. In rare cases you might encounter real old-school VAN that you need to connect to using NSF or Samba connection via Intranet (internal network).

If you imagine that VAN is just a virtual directory (folder) somewhere on the Internet that you can access to move EDI files it will be much easier to understand what you need to make it work. If you can move files in your local computer directory with ease, you can setup similar process for remote directory.

Steps to connect to your trading partner's EDI mailbox:

1. Find out what communication protocol is used.
   You might encounter trading partners that have limited understanding what they have running as EDI service. Typical response might be: "We use VAN". That is not specific enough. You need to find out what communication protocol

they use: FTP, FTP over SSL, AS2, HTTPS, etc. Based on protocol you will know what software you need. Software has to support that specific protocol. Extreme Processing supports some of the protocols and you might not need any other additional software to make it work.

2. Find out what will be the structure of the EDI mailbox.
   There are three typical mailbox structures:
   A) One subdirectory for all incoming and outgoing files.
   B) Two subdirectories. One for incoming and one for outgoing files.
   C) Separate subdirectories for each message type. Example: if you are receiving EDI X12 810 Invoices and sending out EDI X12 997 Acknowledgements and EDI X12 850 Purchase Orders you would have three subdirectories for each message type when you login into EDI mailbox.

3. Find out if there are any file naming patterns.
   If only one subdirectory is used for all message types and both incoming and outgoing files then you will need to separate files based on they specific naming pattern or they actual content. Many EDI systems generate files with specific naming pattern. Typical example is: message type information is embedded in the file name (file called prod837_20090525.txt means that it is production file of EDI X12 837 Healthcare Claim generated on May 25, 2009). Knowing naming convention allows you to separate files without looking at they content.

4. Find out what events will trigger processing.
   If your trading partner sends you files at specific times during the day all you need is to run continues processing, wait for specific hour, connect, look for files, bring them to your systems and process them. If your trading partner sends files at unpredictable times you might need to check EDI mailbox continuously every 10-15 minutes and look for the input files.

5. Find out if you might have concurrency problems.
   When mailbox is used as point of exchange there might be situations when same files are being written when they are being read at the same time. This is especially true for scenarios with continues processing of big files when your process is accessing EDI mailbox files and other process on your trading partner side is accessing same mailbox files. Big files take time to write and read. Reader might end up with incomplete or corrupted file.

   There are number of ways to combat concurrency problems:
   A) If you know at what times your trading partner might be placing files into the mailbox, give your trading partner plenty of time to complete write. If you know that trading partner sends files at 2 PM, do not connect at 2 PM sharp. Wait for 15-20 minutes after 2 PM and connect only then.

   B) If you are writing files into the mailbox and your trading partner is waiting for files with specific naming pattern, create temporary file with naming pattern your trading partner does not expect, then when you finish write, rename the file to that specific naming pattern. Rename operation is fast and does not cause concurrency problems in this case. Your trading partner will only “see” file when it is ready.

   C) Some software products provide special options to fight concurrency problems. Extreme Processing does that too. The way those programs do it is: they read file modification date/time (or file size), then wait for specific relatively short time (20-40 seconds) then check same file modification date/time and if date/time did not change they assume that file is not being written to. This might work and is easy fix because all you need to do is enable this feature. The bad side is that it might not work in all operating systems; it also makes read of files slower because software has to wait and check if file continues to change.

   D) Rare solution that counters concurrency problems you might find on older systems is based on trigger file. Let say trading partner is sending you a file. They system writes EDI file into the mailbox subdirectory and once write is complete another file is created with different name. Second file may be empty and simply acts as a flag that write is complete. Your system should wait for trigger file to appear and only then start reading EDI file.
   This is easily handled with Extreme Processing. You just set two tasks: one that checks for presents of trigger file and one that reads incoming EDI file. Usually once you read EDI file, delete both EDI file and trigger file from mailbox.

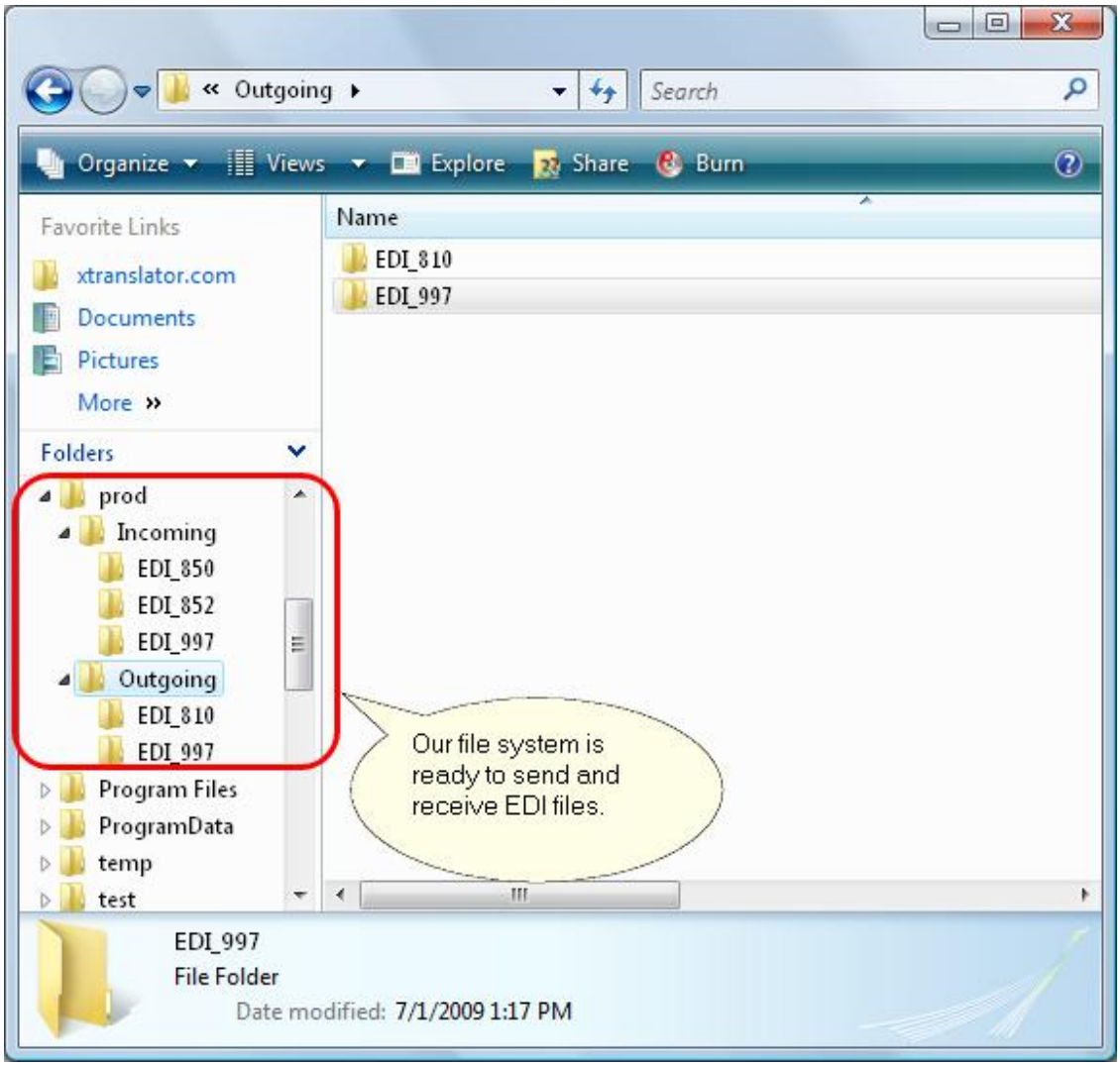
6. Find out possible performance bottlenecks and setup your processing to get better use of resources you have.
   If you do not have dedicated computer for EDI processing and have to run other programs and systems on the same machine make sure at the time when EDI processing is running incoming files other systems are not eating away at valuable CPU time and RAM memory.

Mailbox Automation with Extreme Processing

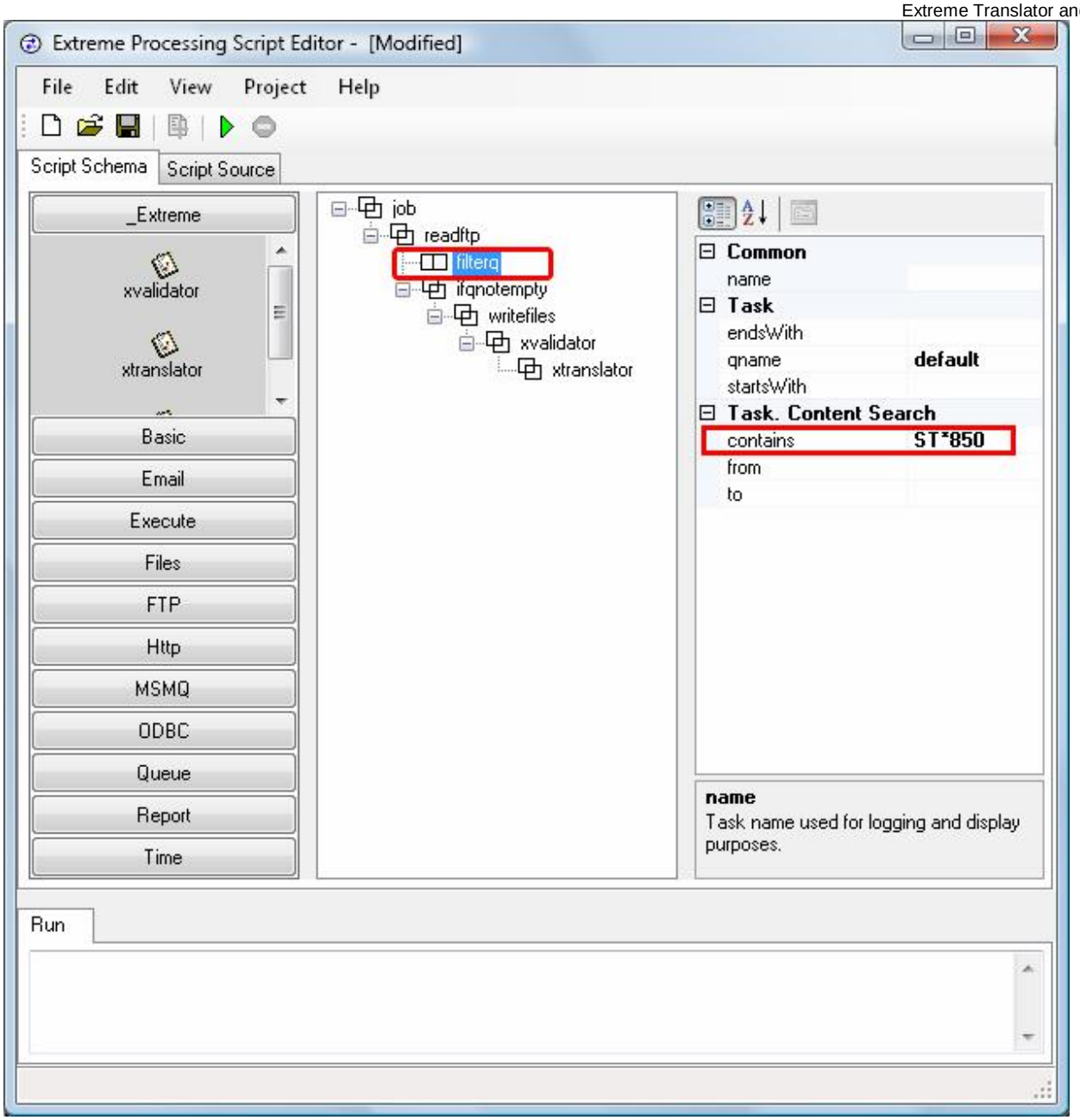
We recommend that you design process flow and file directory structure before creating any processing scripts using Extreme Processing.

Most of topics on use of Extreme Processing product are covered in product's Users Manual. Here we discuss some features and setups related to EDI processing and use of VAN mailboxes.

When you setup communication with VAN mailbox in most cases you can not control how mailbox is setup and what is the file structure on it. Your trading partner or some EDI Clearinghouse sets mailbox for you. But you have control over how files will be handled on your side once they get extracted from the mailbox.
Typical scenario is when mailbox does not have any subdirectories based on file message types and does not distinguish between incoming and outgoing files (no separate sub directories for those). All files that come and go are placed in just one main root directory and when you look at files you suppose to pickup it is not clear what message types those files have unless you look at they content.
In this scenario it is best if you read files from mailbox and move them to your side (your computer or network) and continue process them from your file system. When you are moving files from mailbox route them based on they contents or based on they naming pattern to specific directories on your disk.



We have \prod directory for Production files, then we have two directories for \Incoming and \Outgoing files and then each one of those contains subdirectories for each specific message type.



Script reads files from FTP server, filters them based on they content, and then writes files to process. It looks for EDI 850 Purchase Orders on FTP server, validates them, and runs translator.

You need some kind of router that would execute specific tasks based on the EDI message contained inside the file. Since file names are not descriptive and you can not use patterns to distinguish them only way is to peek inside the file and process it if it contains what your tasks needed.
<filterq> and <keepq> are the tasks that look into contents of files and perform action on them based on the matching patterns.

If file names are descriptive (have obvious naming pattern) you do not need to use <filterq> and <keepq> tasks but use “pattern” properties on <readfiles> or <readftp> or other tasks that read data.