

TIBCO Foresight® Instream®

Instream® API

Software Release 8.7
August 2017

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO Foresight Instream, and TIBCO Foresight Transaction Insight are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

The United States Postal Service holds the copyright in the USPS City State Zip Codes. (c) United States Postal Service 2017.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2010-2017 TIBCO Software Inc. ALL RIGHTS RESERVED.

General Contact Information

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304 USA
Tel: +1 650 846 1000
Fax: +1 650 846 1005

Technical Support

E-mail: support@tibco.com
Web: <https://support.tibco.com>

(Note: Entry to this site requires a username and password. If you do not have one, you can request one. You must have a valid maintenance or support contract to use this site.)

Contents

Introduction

| | |
|--------------------------------------|---|
| Intended Audience | 1 |
| Capabilities..... | 1 |
| Memory-Based Input and Output..... | 1 |
| Checking your Instream Version..... | 2 |
| Checking Validation Results | 2 |
| Paths and Environment Variables..... | 2 |

Java Unified API **3**

| | |
|--|----|
| Requirements..... | 3 |
| Running the Java Demo | 3 |
| Paths and Environment Variables..... | 4 |
| Files..... | 5 |
| Class jNHVInStream | 6 |
| Automatically Choosing a Guideline and APF | 19 |
| RunJavaSample.bat Annotation | 23 |

C++ Unified API **25**

| | |
|-----------------------|----|
| Requirements..... | 25 |
| Files..... | 26 |
| FSInStream Class..... | 27 |

C# API for Windows **37**

| | |
|----------------------------------|----|
| Requirements..... | 37 |
| Setting up your Environment..... | 37 |
| Demo Files | 38 |
| fsInStreamAPI | 38 |

VB.NET API for Windows **47**

| | |
|----------------------------------|----|
| Requirements..... | 47 |
| Setting up your Environment..... | 47 |
| fsInStreamAPI | 47 |

Introduction

Intended Audience

This document is intended for Java, C++, C#, and VB.net developers who are accessing TIBCO Foresight® Instream® validation, Docsplitter, and Response Generator from their applications via API.

Before using this document, familiarize yourself with Instream® validation, which is described in **TIB_fsp-instream_<n.n>_usersguide.pdf**.

If you are using Docsplitter and/or Response Generator, see these documents:

- **TIB_fsp-instream_<n.n>_docsplitter.pdf**
- **TIB_fsp-instream_<n.n>_respngen.pdf**

Capabilities

Using Instream's Java, C++ or C# Unified API, you can load one object and run validation, Docsplitter, Response Generator, and/or Docsplitter from your applications.

Memory-Based Input and Output

| | Validation | Docsplitter | Response Generator | Docsplitter |
|---------------------|------------|-------------|--------------------|-------------|
| Memory input/output | √ | √ | √ | |
| File input/output | √ | √ | √ | √ |

Checking your Instream Version

To use the API described in this document, you will need Instream version 5.3 or later. The VB.net API requires Instream version 5.4.

To check your Instream version, execute **version.bat** in Instream's scripts directory.

Checking Validation Results

Regardless of the API used, the best way to verify validation results is to check the number and types of errors in the Summary results file. Please see **TIB_fsp-instream_<n.n>_usersguide.pdf** for details.

Paths and Environment Variables

| | |
|---------|--|
| Windows | Put Instream's Bin directory in the Path. |
| AIX | export FSINSTREAMINI=/<INSTREAM_ROOT>/bin export LIBPATH=/<INSTREAM_ROOT>/bin:\$LIBPATH |
| Sun | FSINSTREAMINI=<INSTREAM_ROOT>/bin; export FSINSTREAMINI LD_LIBRARY_PATH=/<INSTREAM_ROOT>/bin:\$LD_LIBRARY_PATH; export LD_LIBRARY_PATH |
| HP | export FSINSTREAMINI=/<INSTREAM_ROOT>/bin export SHLIB_PATH=/<INSTREAM_ROOT>/bin:\$SHLIB_PATH |

Java Unified API

Requirements

Please see the readme.txt file that accompanies Instream for Instream requirements information.














You will also need access to persons with knowledge of Java and Instream.

Running the Java Demo

1. Go to Instream's **API\AllProductsAPISampleJava** or **API\JavaSample** directory and compile the .java files by running **Build.bat** or **Build.sh**. You should now have:

JavaApIDemo.class
yourMsgClass.class

2. Run the demonstration by executing **RunJavaSample.bat** or **RunJavaSample.sh**. This file contains several examples.
3. Go to Instream's **Output** directory to see the files created:

| | |
|--|------|
|  JAVAAPIDemo1_Doc_Invalid.txt | 1 KB |
|  JAVAAPIDemo1_Doc_Valid.txt | 1 KB |
|  JAVAAPIDemo1_instream_Result.txt | 5 KB |
|  JAVAAPIDemo3_instream_Result.txt | 5 KB |
|  JAVAAPIDemo4_instream_Result.txt | 1 KB |
|  JAVAAPIDemo5_M_ALL_RespGen.txt | 2 KB |
|  JAVAAPIDemo8_M_ALL_DocSplit.txt | 3 KB |
|  JAVAAPIDemo9_instream_Result.txt | 6 KB |
|  Summary_JAVAAPIDemo1_instream_Result.txt | 1 KB |
|  Summary_JAVAAPIDemo3_instream_Result.txt | 1 KB |
|  Summary_JAVAAPIDemo4_instream_Result.txt | 1 KB |
|  Summary_JAVAAPIDemo9_instream_Result.txt | 1 KB |
|  JAVAAPIDemo1_Doc_Report.xml | 2 KB |

Paths and Environment Variables

Path

Include Instream's **Bin** directory in your path.

classpath

You must put the Instream Java directory in your **classpath**:

```
-classpath "%InStreamRoot%\Java\instreamapi.jar."
```

To set up the environment variable **classpath**, see this example in RunJavaSample.bat:

```
java -classpath ".;%InStreamRoot%\Java\instreamapi.jar" JavaApIDemo
```

↑

①

↑

②

↑

③

- ① Within the quotation marks, the location of JavaAPIDemo.class is identified as “.” – in other words, the current directory from which the batch file is being run.
- ② After the “;” separator, it gives the path and filename of TIBCO Foresight's jar file.
- ③ It then gives the name of the class file being run.

Environment Variable

If you are using an Instream API, we suggest that you use the environment variable FS_INSTREAMUIDDIR to point to a directory where you have write permission.

Files

| Instream's API\AllProductsAPISampleJava or API\JavaSample directory | |
|---|---|
| JavaApIDemo.java | Source code that you can edit and then compile into a.class file. |
| Build.bat | Batch file that compiles JavaApIDemo.java into JavaApIDemo.class and yourMsgClass.java into yourMsgClass.class. |
| RunJavaSample.bat | Batch file that sets up and executes JavaApIDemo.class, which creates validation, Docsplitter, and Response Generator output in Instream's Output directory. See page 19 for details. |
| yourMsgClass.java | Example of how to read a detail and summary record from memory and then notify Instream that validation should continue. |
| Instream's Java directory | |
| instreamapi.jar | TIBCO Foresight's jar file for the unified Java API. |
| Instream's Java\docs\com\foresight\jInStreamAPI directory | |
| jNHVInStream.html | Additional documentation for Class jNHVInStream. |

The other directories under Instream's API directory are for the C++, C#, and VB APIs.

If you plan to change the sample code, copy the project to a different directory. Sample files get reinstalled with each release.

Class jNHVInStream

Import the TIBCO Foresight Java interface at the beginning of your program, like this:

```
import com.foresight.jinstreamapi.*;
```

| Call or Member | Example |
|--|---|
| avoidENV See separator on page 16 . | |
| DocumentType Specifies that the data is XML. Otherwise, a value of "X12" is assumed – which includes all types of EDI that can be validated by Instream. (XML is Windows only) | instObj.DocumentType = "XML"; |
| DS_EdiOutputOpt Specifies that the Docsplitter valid and invalid files are to be output to memory. The only setting is: OUTPUTBYMEMORY OUTPUTBYFILE | instObj.DS_EdiOutputOpt=instObj.OUTPUTBYMEMORY; |
| DS_Options Docsplitter options controlling contents of the report, or disabling the report; one of these: DS_ReportValidOnly DS_ReportInvalidOnly DS_ReportBoth DS_ReportDisable | instObj.DS_Options=instObj.DS_ReportBoth instObj.DS_Options = instObj.DS_ReportDisable |
| DS_ReportInvalidFile Gives the path and filename of the Docsplitter invalid EDI file. | instObj.DS_ReportInvalidFile=OUTPUTDIR+"My_InValid.txt"; |
| DS_ReportValidFile Gives the path and filename of the Docsplitter valid EDI file. | instObj.DS_ReportValidFile=OUTPUTDIR+"My_Valid.txt"; |
| DS_profile Path and filename to a Docsplitter INI file. | public String DS_profile = "NONE"; |

| Call or Member | Example |
|--|---|
| DS_ReportDebugOpt TIBCO Foresight internal use only 0 turns off debug Docsplitter messages 1 turns on debug Docsplitter messages | instObj.DS_ReportDebugOpt=1 |
| DS_ReportFile (file output only) Gives the path and filename of the Docsplitter report. It will contain XML if the filename ends with .xml . It will be a delimited report if the filename ends with .csv . | instObj.DS_ReportFile=OUTPUTDIR+"My_Report.xml"; |
| DS_reportdebug Set to y or n to turn Docsplitter debug messages on or off. | |
| DS_TradingPartnerAutomation Path and filename of a trading partner automation setup (.csv) file that will select a Docsplitter INI file (see TIB_fsp-instream_<n.n>_tpa.pdf). If the Docsplitter INI file contains content splitting information, you will also need to call <code>Doc_getNextInvalidDocument()</code> or <code>Doc_getNextValidDocument()</code> to get all Docsplitter output files. See example 10 in JavaAplDemo.java. | instObj.DS_TradingPartnerAutomation = INPUTDIR + "SampleTPA_DS_RG.csv"; |
| updatemessage This contains information about each split created during content-based Docsplitting: type shows whether the split contained valid or invalid data: 1 for valid, or 2 for invalid data the data in a split index the index of that split – a counter showing which split it is | Example: Assume that Docsplitter split input data into three parts: ISA-IEA - valid ISA-IEA - valid ISA-IEA - invalid You will get three UpdateMessages: 1, data in first ISA-IEA, 1 <i>(the last value shows this is the 1st valid split)</i> 1, data in second ISA-IEA, 2 <i>(the last value shows this is the 2nd valid split)</i> 2, data in third ISA-IEA, 1 <i>(the last value shows this is the 1st invalid split)</i> |
| Doc_getNextInvalidDocument() Doc_getNextValidDocument() Gets all valid and invalid files created by content-based splitting. | String _tstring = instObj.Doc_getNextValidDocument(); |

| Call or Member | Example |
|---|--|
| DS_InvalidEdiOutputStr (memory output only) Memory buffer for the Docsplitter invalid EDI output. | <pre>if(instObj.DS_InvalidEdiOutputStr.length() > 0){ f.write(instObj.DS_InvalidEdiOutputStr.getBytes("UTF8")); }</pre> |
| DS_ReportFormat (Required for memory output; not used for file output) Specifies the format of the Docsplitter report, one of these: XMLFormat (default) DelimitedFormat XMLGenericFormat The XML format slows performance more than the CSV format. | <pre>instObj.DS_ReportFormat=instObj.XMLFormat;</pre> |
| DS_ReportOutputOpt Required for Docsplitter Specifies that the Docsplitter report is to be output to memory or file. Settings can be: OUTPUTBYFILE OUTPUTBYMEMORY | <pre>instObj.DS_ReportOutputOpt=instObj.OUTPUTBYMEMORY;</pre> |
| DS_ReportOutputStr (memory output only) Memory buffer for the Docsplitter report. | <pre>if(instObj.DS_ReportOutputStr.length() > 0){ f.write(instObj.DS_ReportOutputStr.getBytes("UTF8")); }</pre> |
| DS_ValidEdiOutputStr (memory output only) Memory buffer for the Docsplitter valid EDI output. | <pre>if(instObj.DS_ValidEdiOutputStr.length() > 0){ f.write(instObj.DS_ValidEdiOutputStr.getBytes("UTF8")); }</pre> |
| DX_EDInputSourceOption Sets Dataswapper EDI input to memory or file; used when Docsplitter is being used in the mode that calls Instream. Currently, Dataswapper can output to file, but not to memory. It can input from memory if it calls Instream, but not if it accepts validation detail results as input. This can be set to: INPUTBYMEMORY INPUTBYFILE | <pre>myInStream.DX_EDInputSourceOption=myInStream.INPUTBYMEMORY;</pre> |

| Call or Member | Example |
|---|--|
| DX_EDIOutputFile Points to Dataswapper's EDI output file. | myInStream.DX_EDIOutputFile="c:\abc.txt"; |
| DX_EDIOutputOption Sets Dataswapper's EDI output option. This can be set to: OUTPUTBYMEMORY | instObj.DX_EDIOutputOption =instObj.OUTPUTBYMEMORY; |
| getDX_output Requests Dataswapper's EDI output be returned in memory. | (instObj.getDX_output()); |
| getDXReport Requests Dataswapper's report output be returned in memory. | (instObj.getDXReport()); |
| DX_Profile Points to Dataswapper's setup file. | myInStream.DX_EDIOutputFile="c:\Dswap.ini"; |
| DX_ReportFile Points to Dataswapper's report output file. | myInStream.DX_ReportFile="c:\abc.rpt"; |
| flag Describes whether the input and output is by memory or file. Set this flag before you call fshvi(). INPUTBYFILE - the inputFile parameter contains a file name. OUTPUTBYFILE - the outputFile parameter contains a file name. INPUTBYMEMORY – (Instream validation only) the input EDI data is located in memory. The inputMemBuf parameter must also be set to point to the input buffer as described below. OUTPUTBYMEMORY - (Instream validation only) the detail and summary output are to go to memory. In addition to setting this option, the routines updatedataDetail and updatedataSummary must be overridden to perform the actual update to memory. | instObj.flag = instObj.INPUTBYFILE instObj.OUTPUTBYFILE <i>One input and one output flag may be combined via an "or", as in:</i> flag=(INPUTBYMEMORY OUTPUTBYFILE); |

| Call or Member | Example |
|--|---|
| <p>RG_TradingPartnerAutomation</p> <p>Path and filename of a trading partner automation setup (.csv) file that will select a Docsplitter INI file (see TIB_fsp-instream_<n.n>_tpa.pdf).</p> <p>If the Docsplitter INI file contains content splitting information, you will also need to call <code>Doc_getNextInvalidDocument()</code> or <code>Doc_getNextValidDocument()</code> to get all Docsplitter output files.</p> <p>See example 12 in <code>JavaAplDemo.java</code></p> | <pre>instObj.RG_TradingPartnerAutomation=INPUTDIR + "SampleTPA_DS_RG.csv";</pre> |
| <p>FSDOCUMENTONLY</p> <p>This says that the EDI has no enveloping, or its enveloping should be ignored. It will be processed with the ISA and GS definitions specified in:</p> <p>envlopISA</p> <p>envGS</p> <p>FSDOCUMENTONLY, envlopISA, and envGS must <i>all</i> be present if you want the enveloping to be processed this way.</p> <p>Another method for document-only processing is to use <code>avoidENV</code> and <code>separator</code> (see page 16).</p> | <pre>static String envlopISA="ISA*00* *00* *01*9012345720000 *01*9088877320000 *020108*1042*U*00200*000000001*0*T*!"; static String envGS="GS*HP*901234572000*908887732000*20020108*1 615*1*X*004010X091!"; instObj.flag = instObj.FSDOCUMENTONLY;</pre> |
| <p>FSHVINSTREAMROOT</p> <p>By default, the Instream Java API will read MAINDIR from registry ([HKEY_LOCAL_MACHINE, "SOFTWARE\Foresight\Instream"]) as Instream's root directory.</p> <p>It then reads \$dir.ini from MAINDIR\bin.</p> <p>But, you also can use environment variable "FSHVINSTREAMROOT" to overwrite the root directory.</p> <p>(Windows only)</p> | |
| <p>fsInStream</p> <p>Java native interface that calls <code>jHVInStream.dll</code>.</p> | <pre>public int fsInStream(java.lang.String inputFile, java.lang.String outputErrorFile, java.lang.String guideline, java.lang.String errorLevelFile, java.lang.String proFileOptions, java.lang.String inputMemBuf, int flag)</pre> |

| Call or Member | Example |
|--|--|
| fshvi <p>The procedure that actually runs validation, Docsplitter, Response Generator and/or Docsplitter, based on the settings in HVINSTREAM_PROCEDURE (below).</p> <p>Return codes are listed in the user manual for each program (e.g. TIB_fsp-instream_<n.n>-respgen.pdf, etc.).</p> <p>If a code is returned indicating an error, the function <code>getErrorMessage</code> can be used to get a detailed description of the error.</p> | <pre>if(instObj.fshvi()!=100){ System.out.println(instObj.getErrorMessage()); }else{</pre> |
| fs997Report fs999Report fs824Report fs277Report fsContrlReport (EDIFACT only) fsTA1Report fsTextReport <p>(memory output only)</p> <p>If set to true, these create the specific types of Response Generator output in memory.</p> <p>The names of the Response Generator memory buffers are set with <code>OutputBufxxx</code> (see page 14).</p> | <pre>instObj.fs824Report=true; instObj.fsTA1Report=false;</pre> <p>Since <code>fs824Report</code> is set to true, this example creates 824 output in memory.</p> <p>It does not create TA1 output.</p> |
| FSUSINGFSERVER <p>TIBCO Foresight internal use only</p> | |
| getVersion <p>Returns the version of Instream being used.</p> <p>Example: 8 . 7 . 0</p> | <pre>String getVersion(); jNHVInStream instObj = new yourMsgClass(); instObj.fshvi() System.out.println("HVInStream Version = "+instObj.getVersion());</pre> |

| Call or Member | Example |
|---|---|
| <p>InStream_partnerautomation Sets the TPA path for validation. See example 12 in JavaApIDemo.java. If TPA fails due to a missing GS01 or GS08, Instream creates a TA1 file. If an output file name was specified, the TA1 file is named accordingly (e.g., <i>filename.txt.TA1</i>). If memory output is specified, use the OutputBufTA1 call to access the information.</p> | <pre>instObj.InStream_partnerautomation="../bin/SampleTPA_DS_RG.csv"; instObj.OutputBufTA1.getBytes("UTF8");</pre> |
| <p>HVINSTREAM_PROCEDURE Specifies whether fshvi will run Instream validation, Docsplitter, Response Generator and/or Dataswapper</p> <p>INSTREAM_ONLY Run Instream validation only</p> <p>INSTREAM_SPLITTER Run Instream validation and then Docsplitter</p> <p>SPLITTER_ONLY Run Docsplitter only</p> <p>INSTREAM_RESPGEN Run Instream validation and then Response Generator</p> <p>RESPGEN_ONLY Run Response Generator only</p> <p>DATASUBSTITUTE_ONLY Run Dataswapper only</p> <p>INSTREAM_DATASUBSTITUTE Run Instream validation and then Dataswapper</p> <p>FSHV_ALL Run Instream validation, Docsplitter, Response Generator and Dataswapper</p> <p>void_init Reset all variables to the default settings.</p> <p>StopValidationWhenENVHasError Specifies if processing should continue (set to false) or stop (set to true) if an ENV (ISA/GS UNB/UNG) error is encountered. The error is reported in the DTL output in either case.</p> | <pre>instObj.HVINSTREAM_PROCEDURE=instObj.; instObj.HVINSTREAM_PROCEDURE=instObj. INSTREAM_ONLY; instObj.HVINSTREAM_PROCEDURE=instObj. INSTREAM_SPLITTER; instObj.HVINSTREAM_PROCEDURE=instObj. SPLITTER_ONLY; instObj.HVINSTREAM_PROCEDURE=instObj. INSTREAM_RESPGEN; instObj.HVINSTREAM_PROCEDURE=instObj. RESPGEN_ONLY; instObj.HVINSTREAM_PROCEDURE=instObj. DATASUBSTITUTE_ONLY; instObj.HVINSTREAM_PROCEDURE=instObj. INSTREAM_ DATASUBSTITUTE; instObj.HVINSTREAM_PROCEDURE=instObj. FSHV_ALL; or instObj.HVINSTREAM_PROCEDURE=instObj.; instObj.HVINSTREAM_PROCEDURE=instObj.void_init; instObj.HVINSTREAM_PROCEDURE=instObj.StopValidation WhenENVHasError=true;</pre> |

| Call or Member | Example |
|--|---|
| getErrorMessage Returns the text of the last error encountered by Instream validation. | <pre>if(instObj.fshvi()!=100){ System.out.println(instObj.getErrorMessage()); }</pre> |
| getETYPE_type Returns count of errors of a particular type: TypeZeroCount SyntaxCount TypeTwoCount SituationCount CodeSetCount ProductCount PayerCount PartnerCount | <pre>System.out.println(instObj.getETYPE_CodeSetCount());</pre> |
| getSVRTY_severity (File-based output only) Returns count of errors of a particular severity: IgnoreCount InfoCount WarningCount ErrorCount FatalCount User1Count User2Count | <pre>System.out.println(instObj.getSVRTY_ErrorCount());</pre> |
| guideline Gives the name of the guideline to be used for validation or input into another program such as Docsplitter. If omitted, Instream uses the first guideline encountered that matches the data's Version/Release Identifier code (Segment GS Element 08). If you are using Docsplitter, be sure it is a GuidelinePlus or is based on a GuidelinePlus (one that starts with PD). | <pre>instObj.guideline="PDSA835"; myInStream.guideline="PDSA835";</pre> |

| Call or Member | Example |
|--|--|
| inputFile <p>When using INPUTBYFILE, this gives the full path and name of the EDI data file that will be validated.</p> <p>When using INPUTBYMEMORY, this gives a name or title for the data to be output in the Start Message.</p> | instObj.inputFile=INPUTDIR+"My_Demo.txt"; |
| inputMemBuf <p>Input parameter pointing to memory buffer where input EDI data is to be found. This parameter is required when INPUTBYMEMORY is specified in the flag parameter (see above), and ignored otherwise.</p> | instObj.inputMemBuf = new String(b,0,n); |
| InStreamOutputXmlFormat <p>Set to "true" if you want an XML-format validation report.</p> <p>Set to "false" if you want a delimited format validation report.</p> | |
| origFileInfo <p>Specifies the month, day, year, hour, minute, second, size, and path of the original EDI file.</p> | instObj.origFileInfo="02/14/05 14:55:19 2032 C:/HVInStream/DemoData/835-DEMO1.TXT"; |
| OutputBuf997 OutputBuf999 OutputBuf277 OutputBuf824 OutputBufTA1 OutputBufCus Get997Report Get999Report GetContrlReport (EDIFACT only) <p>(Response Generator memory output only)</p> <p>These are the buffer names for the Response Generator output.</p> | f.write(instObj.OutputBuf824.getBytes("UTF8")); f.write(instObj.OutputBufTA1.getBytes("UTF8")); |

| Call or Member | Example |
|---|--|
| outputErrorFile <p>Gives the name of the validation detail results file.</p> <p>When using OUTPUTBYFILE, this specifies the full path name to the output detail file.</p> <p>When using OUTPUTBYMEMORY, this is a file name to be used if needed due to failure of memory output.</p> | <pre>instObj.outputErrorFile=OUTPUTDIR+"My_Result.txt";</pre> |
| OutputXmlFormat TIBCO Foresight Internal use only | |
| profileOptions <p>Gives the name of the validation profile file.</p> <p>Defaults to \$FSDEFLT.APF.</p> | <pre>instObj.profileOptions="users.apf";</pre> |
| PROFILEUPDATED TIBCO Foresight internal use only | |
| respGenOutputFlag <p>Specifies that Response Generator output goes to memory or to a file:</p> <p>OUTPUTBYMEMORY OUTPUTBYFILE</p> | <pre>instObj.respGenOutputFlag=instObj.OUTPUTBYMEMORY;</pre> |
| RG_Contrl <p>Directs RG_Contrl output to memory or to a file.</p> | <p>File output :</p> <pre>instObj.RG_Contrl = OUTPUTDIR + "JAVAAPISample1_RespGen_Control.txt";</pre> <p>Memory Output :</p> <pre>instObj.fsContrlReport = true; String msg = instObj.GetContrlReport();</pre> |
| RG_Options <p>Response Generator options, which can include any options mentioned in TIB_fsp-instream_<n.n>-respgen.pdf, except -TPA.</p> | <pre>instObj.RG_Options="-ge -y"</pre> <p><i>This example specifies that group enveloping is to be included in the response documents and that Response Generator can overwrite files if they already exist.</i></p> <p>Important: When specifying Response Generator options via the API, leave no space between the option and its value. For example, use -er3 instead of -er 3. Spaces may cause the API response generator call to fail.</p> |

| Call or Member | Example |
|--|--|
| RG_Rep277 RG_Rep824 RG_Rep997 RG_Rep999 RG_TA1 <p>Gives the name of the EDI file(s) to be created by Response Generator.</p> | <pre>instObj.RG_Rep997=OUTPUTDIR+"My_Demo_997.txt" instObj.RG_Rep277 = OUTPUTDIR + "JAVAAPISample1_RespGen_277.txt"; instObj.RG_Rep999 = OUTPUTDIR + "JAVAAPISample1_RespGen_999.txt"; instObj.RG_TA1 = OUTPUTDIR + "JAVAAPISample1_RespGen_TA1.txt";</pre> |
| RG_repText RG_Reptemp <p>RG_repText gives the name of a custom text report to be created by Response Generator.</p> <p>RG_Reptemp gives the name of the template that controls the format of the custom text report.</p> <p>If you use one of these, use both.</p> | <pre>instObj.RG_repText=OUTPUTDIR+"RG_text_rpt.txt"; instObj.RG_Reptemp=INPUTDIR+"Template.txt";</pre> |
| separator <p>For “document-only” validation - when the data does not have ISA or GS enveloping, or when the enveloping is to be ignored. Lists the segment, element, and composite separators.</p> <p>It must appear BEFORE avoidENV.</p> <p>Separators can be in any of these formats:</p> <p>Integer example: 29,30,31</p> <p>hexadecimal example: 0x1E,0x1F,0x1D</p> <p>character example: ~!*</p> <p>avoidENV=“NONE”</p> <p>TIBCO Foresight internal use only</p> <p>This specifies that validation is to ignore ISA and GS enveloping and use the separators provided with separator.</p> <p>Another method for document-only processing is to use FSDOCUMENTONLY (see page 10).</p> | <pre>instObj.separator=~*.*; instObj.avoidENV=true;</pre> |

| Call or Member | Example |
|---|--|
| <p>updatedataDetail</p> <p>This procedure writes detailed validation output to memory when the OUTPUTBYMEMORY flag is set. This is done one record at a time.</p> <p>You must override this routine to perform the actual output as desired. The default version of this routine just sends the detail message to standard output.</p> <p>You can then have your application evaluate the record and either terminate validation or continue validation.</p> <p>If you want to terminate validation, return the constant STOPHVINSTREAM (or 500).</p> <p>If you want to continue validation, return the constant CONTHVINSTREAM (or 510).</p> | <pre>public int updatedataDetail(String detail){ System.out.print(detail); return(CONTHVINSTREAM); }</pre> |
| <p>updatedataSummary</p> <p>This procedure writes summary validation output to memory when the OUTPUTBYMEMORY flag is set. This is done each time a transaction set ends in the detail output.</p> <p>You must override this routine to perform the actual output as desired. The default version of this routine just sends the detail message to standard output.</p> <p>You can then have your application evaluate the output and either terminate validation or continue validation.</p> <p>If you want to terminate validation, return the constant STOPHVINSTREAM (or 500).</p> <p>If you want to continue validation, return the constant CONTHVINSTREAM (or 510).</p> | <pre>public int updatedataSummary(String summary){ return (CONTHVINSTREAM); }</pre> |

| Call or Member | Example |
|--|--|
| <p>updateMessage</p> <p>This procedure is responsible for writing Document Splitter and Data Exchange values output to memory.</p> <p>Where type = <n></p> <p>1 - Memory output, value1 = DocSplitter Valid document, value2 = Content name for Content-Based splitting.</p> <p>2 - Memory output, value1 = DocSplitter Invalid document, value2=Content name for Content-Based splitting.</p> <p>3 - Memory output, value1 = DocSplitter Report, value2 = Not used.</p> <p>4 - Memory output, value1 = Data exchange output, value2 = Not used.</p> <p>5 - Memory output, value1 = Data exchange Report, value2 = Not used.</p> <p>6 - File output, value1 = Content-Based splitting valid document name, value2 = Not used.</p> <p>7 - File output, value1 = Content-Based splitting invalid document name, value2 = Not used.</p> | <pre>int updateMessage(int type, java.lang.String value1, java.lang.String value2)</pre> |
| <p>userMessage</p> <p>Inserts your free-form text in a validation detail file GEN record with number 15078.</p> <p>Example GEN record (assume your text was "Sock 2"):</p> <pre>GEN 015078 1 0Sock 2</pre> <p>This is for your own use. TIBCO Foresight® Transaction Insight® can display it.</p> <p>It also provides a role in trading partner automation. Please see When the Guideline and APF cannot be Identified on page 21.</p> | |

| Call or Member | Example |
|---|-----------------------------------|
| UseUpdateAnalysis This lets the calling program pick the guideline and validation profile. You will need to do two things to set this up: <ol style="list-style-type: none"> 1. Set the UseUpdateAnalysis flag to true before calling fshvi. 2. Modify the calling program to select a guideline. Please see UseUpdateAnalysis below for details. | instObj.UseUpdateAnalysis = true; |

Automatically Choosing a Guideline and APF

This feature lets you choose a guideline, and optionally a profile, based on the contents of the data.

UseUpdateAnalysis

For an example, please see yourMsgClass.java in TIBCO Foresight's API\AllProductsAPISampleJava directory.

X12 - values in ISA or GS

To make the guideline and profile decision for X12 data, based on values in the ISA or GS:

1. Do not specify a guideline.
2. Set the UseUpdateAnalysis flag to true.
3. At each GS, the calling program will receive UpdateAnalysis with the <interchange> and <FunctionalGroup> data shown below.

The <Version> tag will contain the X12 version .

4. The calling program sends back **this.guideline** and **this.profileOptions** variables:

```

this.guideline = "B41A837I";
retVal |= this.STANDARDUPDATED;

    /** Set apf file to use *****/
    this.profileOptions = "XXXX.apf";
    retVal |= this.PROFILEUPDATED;

```

Flat File, XML and X12 transaction set data

To make the guideline and profile decision for XML or flat file data, or for X12 data in the transaction set rather than in the enveloping:

1. Use **guideline** to specify a content-based guideline (see **TIB_fsp-instream_<n.n>-tpa.pdf**). This will contain an IdentifierLookup business rule and a DSR mark to show locations where a guideline or profile switch may occur.
2. Set the UseUpdateAnalysis flag to true.
3. During validation, at each DSR location, the calling program will receive UpdateAnalysis containing the <Variables> element information shown below. The <Interchange> and <FunctionalGroup> elements will be empty.

The <Version> tag will contain Flat or XML .

4. The calling program sends back **this.guideline** and **this.profileOptions** variables:

```
this.guideline = "B41A837I";
retVal |= this.STANDARDUPDATED;
this.profileOptions = "XXXX.apf";
retVal |= this.PROFILEUPDATED;
```

Example XML sent to calling program by UpdateAnalysis

```
<?xml version="1.0" encoding="UTF-8"?>
  <Info>
    <Interchange>
      <SenderQualifier>01</SenderQualifier>
      <Sender>9012345720000</Sender>
      <ReceiverQualifier>01</ReceiverQualifier>
      <Receiver>9088877320000</Receiver>
      <Date>020111</Date>
      <Time>1212</Time>
      <CtlNo>000000001</CtlNo>
    </Interchange>
    <FunctionalGroup>
      <ID>HC</ID>
      <Sender>901234572000</Sender>
      <Receiver>908887732000</Receiver>
      <Date>20020111</Date>
      <Time>1615</Time>
      <CtlNo>1</CtlNo>
      <Version>004010X096A1</Version>
    </FunctionalGroup>
  </Info>
```



```

<Variables>
<variable index="1" segment="CS" element="5">Topcat Co.</variable>
<variable index="2" segment="" element=""></variable>
</Variables>
</Info>

```

When the Guideline and APF cannot be Identified

With `userMessage`, you can handle several trading partner automation conditions where the guideline and APF cannot be identified from the values in the enveloping.

The `userMessage` provides an error number that Instream uses to:

- Identifies the envelope condition that caused the partner automation to fail
- Send text that you specified
- Send return code 191
- For some error numbers, create TA1, 997, or 999 response documents

The format is:

```
userMessage = "error_code, message";
```

Where:

| | |
|-------------------|---|
| <i>error_code</i> | One of the error codes in the Error Messages Table below. This will determine whether Instream creates TA1, 997, or 999 response documents, and what error message is returned. |
| <i>message</i> | An error message worded by you. |

Example

```
userMessage = "10039, Interchange version is found but no receiver
qualifier and ID matches";
```

Referring to the table below, we see that error number 10039 automatically creates a TA1 containing 022. The specified text is returned ("Interchange version is found but no receiver qualifier and ID matches"), along with return code 191.

Please see **Example 12. Trading Partner Automation** in `javaAPIDemo.java`, in Instream's `API\AllProductsAPISampleJava` directory.

| Error Messages Table | | |
|----------------------|---|------------------------------|
| Error code | Meaning | Responses created |
| 10037 | Interchange version is not found | None |
| 10038 | Interchange version is found but the sender ID/Qualifier pair is not found for any partner with that interchange version | None |
| 10039 | Interchange version is found but receiver ID/Qualifier pair is not found for any trading partner with that interchange version | TA1 with 022 |
| 10040 | Interchange version and ID/Qual for both sender and receiver are found, but the interchange segment has other validation errors | TA1 with 016 |
| 10041 | Group version is not found | TA1 accept |
| 10042 | Group version is found but the sender-receiver pair is not found | TA1 accept 997/999 reject |
| 10043 | Group version, group sender, and group receiver are all found, but the group segment contains other errors | TA1 accept 997/999 reject |
| 10044 | Transaction version is not found | TA1 accept 997/999 reject |

RunJavaSample.bat Annotation

This file sets up paths and runs JavaApiDemo.class.

| From File | Explanation |
|--|---|
| set InStreamRoot=C:\Program Files\HIPAA Validator InStream | InStreamRoot points to high-level Instream directory. |
| set InputDir=%InStreamRoot%\DemoData set OutputDir=%InStreamRoot%\Output | InputDir and OutputDir are directories under InStreamRoot. |
| rem Instream Bin directory must be in path in order for Java to find jHVInStream.dll rem If it is not in the path, we add it here echo>__tmp__ %PATH% find>nul: /I "%InStreamRoot%\Bin" __tmp__ if ERRORLEVEL 1 set PATH=%PATH%;%InStreamRoot%\Bin del>nul: __tmp__ | If Instream's Bin directory is not in the path, it is located and added to the path. |
| @echo Using Java to run 835-DEMO1.TXT data through Instream... java -classpath ".;%InStreamRoot%\Java\instreamapi.jar" JavaApiDemo The dot just after the opening quotation mark says the path to JavaApiDemo.class is the same directory as the batch file. The semi-colon after it is a separator. | Run JavaApiDemo.class. To do this, you: <ul style="list-style-type: none">• Identify the name and location of TIBCO Foresight's jar file. It's in InStream's Java directory: <u>%InStreamRoot%\Java\instreamapi.jar</u>• Identify the name and location of our input file (JavaApiDemo.class). It is in the same directory as the batch file you run to execute the program (RunJavaSample.bat). |
| @echo. @echo Finished - Results can be found in %OutputDir% pause | Displays message about where to find output. |

C++ Unified API

Requirements

Please see the readme.txt file that accompanies Instream for Instream requirements information.

You will also need:

- Access to persons with knowledge of C++ and Instream.
- Compiler:

Windows Microsoft Visual C++ version 2008 or later

UNIX GCC version:

| | |
|-------------|-------|
| AIX | 4.1.1 |
| HP | 3.2 |
| Sun Solaris | 3.3.2 |
| Linux | 3.4.5 |

Files

| Instream's API\InStreamAPISample or API\C++Sample Directory | |
|--|---|
| File | Location / Purpose |
| InStreamAPISample.dsw | (Windows only) Microsoft Developer Studio Workspace File. |
| InStreamAPISample.dsp | (Windows only) Microsoft Developer Studio Project File. |
| InStreamAPISample.h | Sample header file that funnels the detail results and summary results to your program one line at a time. |
| InStreamAPISample.cpp | Sample source code that you can edit. |
| Required Files | |
| FSInStream.h (Windows) FSInStreamunix.h , (UNIX) | <p>Instream's API \ Include directory</p> <p>Defines the classes, procedures, and constants needed to interface to the Instream Validation Engine. Include it in your C++ source file. The main class needed (and contained in this file) is FSInStream, which is described in the next section.</p> <p>Note: Do not change this file. This object is passed to the validation engine and may be changed with next installation.</p> |
| HVInStream.dll (Windows) libHVInStream.so/sl (UNIX) | <p>Instream's Bin directory</p> <p>The Instream dynamic-link library.</p> <p>Include the Bin directory is the system's Path.</p> |
| FSInStream.cpp (Windows) FSInStreamunix.cpp (UNIX) | <p>Instream's API \ Source directory</p> <p>File that actually calls the validation engine. Add FSInStream to you own project that calls the Validation engine.</p> <p>Note: Do not change this file. This object is passed to the validation engine and may be changed with next installation.</p> |

If you plan to make changes to the sample code, copy the project to a different directory. Sample files get reinstalled with each release.

FSInStream Class

The **FSInStream** class is the link between your C/C++ application and the Instream Validation Engine Library. An object of the **FSInStream** class is initiated, its input parameters initialized to the appropriate files, options, etc. and then the **Analyze** procedure is called to perform the validation.

| Methods | |
|--|--|
| Call | Example |
| Analyze Once you have called the necessary Set and Get methods, call the Analyze method to perform the actual validation. Return codes are listed in TIB_fsp-instream_<n.n>_usersguide.pdf . If a return code indicates an error, you can use the function GetLastErrorMessage to get a detailed description of the error. | <pre>int retVal=myInStream.Analyze();</pre> |
| RespGen Once you have called the necessary Set and Get methods, call the RespGen method to perform the actual generation of 997, 999, 824, 277, or custom text reports. Return codes are listed in TIB_fsp-instream_<n.n>_respgen.pdf . | <pre>retVal=myInStream.RespGen();</pre> |
| Split Once you have called the necessary Set and Get methods, call the Split method to perform the actual document splitting. Return codes are listed in TIB_fsp-instream_<n.n>_docsplitter.pdf . | <pre>retVal=myInStream.Split();</pre> |
| Substitution Once you have called the necessary Set methods, call the Substitution method to run Docsplitter. Return codes are listed in TIB_fsp-instream_<n.n>_docsplitter.pdf . | <pre>retVal = myInStream.Substitution();</pre> |

| Methods | |
|--|--|
| Call | Example |
| <p>DocumentLevelOnly</p> <p>This lets you process EDI data that has no enveloping. When set to true, the Interchange and Functional Group envelope is processed with the data in SetIntEnvelope and SetFGEnvelope (below).</p> <p>Both SetIntEnvelope and SetFGEnvelope must be included if you want to use DocumentLevelOnly.</p> <p>If the incoming EDI data contain an Interchange and Functional Group Envelope, its enveloping will always be used regardless of DocumentLevelOnly.</p> <p>Another method for document-only processing is to use m_separator (see page 30).</p> | <pre>//Set Document Level Only Parameters myInStream.DocumentLevelOnly(true); myInStream.SetIntEnvelope("ISA*00* *00* *01*9012345720000 *01*9088877320000 *020108*1042*U*00200*000000001*0*T*!"); myInStream.SetFGEnvelope("GS*HP*901234572000*908 887732000*20020108*1615*1*T*004010X091!");</pre> |
| <p>getDocReport</p> <p>(memory output only)</p> <p>Memory buffer for the Docsplitter report.</p> | <pre>p=myInStream.getDocReport();</pre> |
| <p>getInValidEdi</p> <p>(memory output only)</p> <p>Memory buffer for the Invalid EDI file created by Docsplitter.</p> | <pre>p=myInStream.getInValidEdi();</pre> |
| <p>GetLastErrorMessage</p> <p>Returns the text of the last error encountered by Instream validation.</p> | <pre>if(retVal != FSSUCCESS) printf("%s",myInStream.GetLastErrorMessage());</pre> |
| <p>GetRGOutput997 GetRGOutput999 GetRGOoutput277 GetRGOoutput824 GetRGOoutputTA1 GetRGOoutputReport</p> <p>(Response Generator memory output only)</p> <p>These are the buffer names for the Response Generator output.</p> | <pre>char * myInStream.GetRGOoutput277(); myInStream.m_RGReportItem.rg_999=true; const char *p=myInStream.GetRGOutput999();</pre> |
| <p>getValidEdi</p> <p>(memory output only)</p> <p>Memory buffer for the Valid EDI file created by Docsplitter.</p> | <pre>char * myInStream.getValidEdi();</pre> |

| Methods | |
|---|---|
| Call | Example |
| getVersion Returns the version of Instream being used. Example: 4.2.0 | <pre>char * getVersion(); FSInStream * myObj=new dllDriver(); printf("HVInStream Version = %s",myObj->getVersion());</pre> |
| m_svrty.SUM_ severity (File-based output only) Returns count of errors of a particular severity: IgnoreCount InfoCount WarningCount ErrorCount FatalCount User1Count User2Count | <pre>/** Split the input EDI file, if there are errors */ if(myInStream.m_svrty.SUM_FatalCount > 0 myInStream.m_svrty.SUM_ErrorCount > 0) { ...</pre> |
| m_etype.SUM_ type (File-based output only) Returns count of errors of a particular type: TypeZeroCount SyntaxCount TypeTwoCount SituationCount CodeSetCount ProductCount PayerCount PartnerCount | <pre>if(myInStream.m_etype.SUM_SyntaxCount > 0 myInStream.m_etype.SUM_TypeTwoCount > 0) { ...</pre> |
| m_originalFileInfo.mon m_originalFileInfo.day m_originalFileInfo.year m_originalFileInfo.hour m_originalFileInfo.min m_originalFileInfo.sec m_originalFileInfo.fsize m_originalFileInfo.FileName Specifies the month, day, year, hour, minute, second, size, and path of the original EDI file. | <pre>myInStream.m_originalFileInfo.mon=2; myInStream.m_originalFileInfo.day=25; myInStream.m_originalFileInfo.year=2005; myInStream.m_originalFileInfo.hour=14; myInStream.m_originalFileInfo.min=12; myInStream.m_originalFileInfo.sec=34; myInStream.m_originalFileInfo.fsize = 9999989; strcpy(myInStream.m_originalFileInfo. FileName,"C:\\Temp\\abc.txt");</pre> |

| Methods | |
|--|--|
| Call | Example |
| <p>m_separator</p> <p>For “document-only” validation - when the data does not have ISA or GS enveloping, or when the enveloping is to be ignored.</p> <p>m_separator.seg m_separator.elm m_separator.comp</p> <p>Delimiters and separators for document-only validation. They can be in these formats:</p> <p>integer example: 29</p> <p>hexadecimal example: 0x1E</p> <p>character example: ~</p> <p>m_separator.ignoreENV</p> <p>Validation is to ignore ISA and GS enveloping and use the separators provided with m_separator.seg, m_separator.elm and m_separator.comp.</p> <p>This requires that m_separator.seg, m_separator.elm and m_separator.comp be provided, and it must come AFTER them.</p> <p>Another method for document-only processing is to use DocumentLevelOnly (see page 28).</p> | <pre>myInStream.m_separator.seg='~'; myInStream.m_separator.elm='*'; myInStream.m_separator.comp=': '; myInStream.m_separator.ignoreENV=true;</pre> |
| <p>rg_277 rg_824 rg_997 rg_TA1 rg_text</p> <p>(memory output only)</p> <p>If set to true, these create the specific types of Response Generator output in memory.</p> | <pre>myInStream.m_RGReportItem.rg_824=true; myInStream.m_RGReportItem.rg_TA1=false;</pre> <p>Since RGReportItem.rg is set to true, this example creates 824 output in memory.</p> <p>It does not create TA1 output.</p> |

| Methods | |
|--|--|
| Call | Example |
| setDocReportFormat (memory output only) Specifies the format of the Docsplitter report, one of these: DocXMLFormat DocDelimitedFormat The XML format slows performance more than the CSV format. | <pre>myInStream.setDocReportFormat(DocXMLFormat);</pre> |
| SetDSOptions Docsplitter options to create valid and/or invalid EDI files using one of these: DOC_ReportBoth DOC_ReportInvalidOnly DOC_ReportValidOnly SetDSReportInvalidFile Gives the path and filename of the Docsplitter invalid EDI file. SetDSReportValidFile Gives the path and filename of the Docsplitter valid EDI file. | <pre>myInStream.SetDSOptions(DOC_ReportValidOnly); /** ValidFile file name ** tempString = INSTREAMROOT; tempString += "/Output/DocSplitter_835_Valid.txt"; myInStream.SetDSReportValidFile(tempString.c_str()); /** InValidFile file name ** tempString = INSTREAMROOT; tempString += "/Output/DocSplitter_835_InValid.txt"; myInStream.SetDSReportInvalidFile(tempString.c_str());</pre> |
| SetDSplitEdiOutputOpt Specifies that the Docsplitter valid and invalid EDI is to be output to memory. Settings can be: OutputByMemory OutputByFile | <pre>myInStream.SetDSplitEdiOutputOpt(OutputByMemory);</pre> |
| SetDSplitReportOutputOpt Specifies that the Docsplitter report is to be output to memory. Settings can be: OutputByMemory OutputByFile | <pre>myInStream.SetDSplitReportOutputOpt(OutputByMemory);</pre> |
| SetDSReportDebug 0 turns off debug Docsplitter messages 1 turns on debug Docsplitter messages Default is 0. | <pre>myInStream.SetDSReportDebug();</pre> |

| Methods | |
|--|---|
| Call | Example |
| SetDSReportFile Sets the path and filename of the Docsplitter report. It will contain XML if the filename ends with .xml . It will be a delimited report if the filename ends with .csv . | <pre>tempString = INSTREAMROOT; tempString += "/Output/DocSplitter_835_Report.xml"; myInStream.SetDSReportFile(tempString.c_str());</pre> |
| SetDSubEDIInputSourceOption Sets Docsplitter's EDI input to memory or file; used when Docsplitter is being used in the mode that calls Instream. Currently, Docsplitter can output to file, but not to memory. It can input from memory if it calls Instream, but not if it accepts validation detail results as input. This can be set to: InputByMemory InputByFile | <pre>myInStream.SetDSubEDIInputSourceOption(InputByMemory);</pre> |
| SetDSubEDIOutputFile Points to Docsplitter's EDI output file. Must be set to strEDIOutputFile . | <pre>myInStream.SetDSubEDIOutputFile("c:/output/nEdi.txt");</pre> |
| SetDSubReportFile Points to Docsplitter's report output file. Must be set to strReportFile . | <pre>myInStream.SetDSubReportFile("c:/output/nRep.txt");</pre> |
| SetGuideline Specifies the guideline to use for the validation. If omitted, Instream uses the first guideline encountered that matches the data's Version/Release Identifier code (GS08). If you are using Docsplitter, be sure it is a GuidelinePlus or is based on a GuidelinePlus (one that starts with PD). | <pre>myInStream.SetGuideline("PDSA837I");</pre> |

| Methods | |
|--|--|
| Call | Example |
| <p>SetInputFile</p> <p>When using InputByFile, this contains the full path name to the input EDI data file to be validated.</p> <p>When using InputByMemory, this contains a name or title for the data to be output in the Start Message (see STRT record in TIB_fsp-instream_<n.n>-usersguide.pdf).</p> <p>Be sure the directory already exists for the specified file.</p> | <pre>myInStream.SetInputFile("c:/demo/nEdi.txt");</pre> |
| <p>SetInputMemory</p> <p>Location of the memory buffer for the EDI data being input to Instream.</p> <p>This parameter is required when SetOutputSource is OutputByMemory and is ignored otherwise.</p> | <pre>myInStream.SetInputMemory(pBuffer);</pre> |
| <p>SetInputSourceOption</p> <p>InputByFile – Means the filename specified with SetInputFile is the source of the EDI data.</p> <p>InputByMemory – Must SetInputMemory to the memory location of the EDI data.</p> | <pre>myInStream.SetInputSourceOption(InputByMemory);</pre> |
| <p>SetIntEnvelope and SetFGEnvelope</p> <p>Defines the Interchange and Functional Group Envelopes when DocumentLevelOnly is set to true.</p> <p>If the EDI data contains an Interchange or Functional Group envelope, the ones in the file will be used even if SetIntEnvelope and SetFGEnvelope have been set.</p> | <p>See DocumentLevelOnly example above.</p> |
| <p>SetOutputErrorFile</p> <p>When using OutputByFile, this contains the full path name of the validation detail file to be generated.</p> <p>When using OutputByMemory, this contains a file name in case problems occur and Validator needs a 'last resort' place to put output.</p> | <pre>myInStream.SetOutputErrorFile("path/name");</pre> |

| Methods | |
|---|---|
| Call | Example |
| <p>SetOutputSourceOption</p> <p>OutputByFile – (default) Detail and Summary validation results should go to the file indicated by SetOutputErrorFile. The summary file will have the same name with SUMMARY in front of it.</p> <p>OutputByMemory – The methods DetailMessage and SummaryMessage will get called as an output line is created.</p> <p>Avoid OutputByMemory if you are feeding the validation output into Docsplitter or Response Generator.</p> <p>DetailMessage / SummaryMessage</p> <p>The default version of this routine sends the message to standard output. To perform something different, override these routines in the FSInStream derived object. If you wish to terminate processing from this routine, you can return false.</p> <p>Otherwise, true should be returned to continue processing.</p> | <pre>myInStream.SetOutputSourceOption(OutputByMemory)</pre> |
| <p>SetProfileFile</p> <p>Specifies the name of the profile file to be used for the validation. Please see APF.pdf.</p> <p>If the path is omitted, the Bin directory is assumed.</p> <p>If SetProfileFile is omitted, the validation will use the default profile \$fsdeflt.apf (in Instream's Bin directory).</p> | <pre>myInStream.SetProfileFile("C:\Program Files\HIPAA Validator InStream\Bin\ourusual.apf");</pre> |
| <p>SetRespGenOutputOpt</p> <p>Specifies that Response Generator output goes to memory or file. It can be set to:</p> <p>OUTPUTBYMEMORY OUTPUTBYFILE</p> | <pre>myInStream.SetRespGenOutputOpt(OutputByMemory);</pre> |

| Methods | |
|--|---|
| Call | Example |
| SetRGOptions Response Generator options, which can include any options mentioned in TIB_fsp-instream_<n.n>_respgen.pdf . | <pre>myInStream.SetRGOptions("-ge -y");</pre> <p><i>This example specifies that group enveloping is to be included in the response documents and that Response Generator can overwrite files if they already exist.</i></p> <p>Important: When specifying Response Generator options via the API, leave no space between the option and its value. For example, use -er3 instead of -er 3. Spaces may cause the API response generator call to fail.</p> |
| SetRGRep824File SetRGRep997File SetRGRep277File SetRGRep999File Gives the name of the EDI 824, 997, 277, or 999 to be created by Response Generator. | <pre>myInStream.SetRGRep824File(tempString.c_str());</pre> <pre>myInStream.SetRGRep999File(999_output_filename);</pre> |
| SetRGReptextFile SetRGReptempFile SetRGRepTextFile gives the name of a custom text report to be created by Response Generator. SetRGRepTplFile gives the name of the template file to be used when creating the custom text report. If you use one of these, use both. | <pre>myInStream.SetRGReptextFile("path/name");</pre> <pre>myInStream.SetRGReptempFile("path/name");</pre> |
| SetUserMessage Specifies free-form text to be inserted in a GEN record with number 15078. Example: GEN 015078 1 0Sock 2 This is for your own use. Transaction Insight® can display it. | <pre>myInStream.SetUserMessage("Sock 2 ");</pre> |

C# API for Windows

Requirements

Please see the readme.txt file that accompanies Instream for Instream requirements information.

You will also need:

- Access to persons with knowledge of C# and Instream
- Microsoft Windows. There is no UNIX version of this API
- Microsoft Visual Studio 2003 or later

Setting up your Environment

- Put Instream's Bin directory in your path.
- The DllImport attribute is used to specify the name of the DLL (CSInStreamAPI.dll) that contains the methods that are available in the API. This attribute is set when Instream is installed. There are several examples in the C# sample (i.e., fsInstreamAPI.cs).

The example below references the Instream_info method in the CSInStreamAPI.dll.

```
[ DllImport(@"d:\Foresight\instream\8.7\Bin\CSInStreamAPI.dll",  
  CharSet=CharSet.Ansi, CallingConvention=CallingConvention.StdCall) ]  
public static extern int InStream(UpdateMessage myCallBack, *ref  
  InStream_info *mobj, string edibuf );
```

Note: If you move CSInStreamAPI.dll to a different location, you must change the path name in the values property of the DllImport attribute.

Demo Files

Instream's **API\InStreamCSharpAPISample** or **API\ C#Sample** directory contains a sample project. **csharpDemo.cs** is an example program.

TIBCO Foresight's C# API is in **fsInStreamAPI.cs**. Changes to this file:

- You must customize the Report section at the very end if you want to use memory output.
- If you move CSInStreamAPI.dll, update the path to it in this file.
- Do not change anything else.

fsInStreamAPI

| Methods | Example |
|--|--|
| DocInvalidEdi (memory output only) Memory buffer for the Docsplitter invalid EDI output. | <code>sw.Write(InStreamAPI.fsInStreamAPI.DocInvalidEdi.ToString());</code> |
| DocReport (memory output only) Memory buffer for the Docsplitter output report. | <code>sw.Write(InStreamAPI.fsInStreamAPI.DocReport.ToString());</code> |
| DocValidEdi (memory output only) Memory buffer for the Docsplitter valid EDI output. | <code>sw.Write(InStreamAPI.fsInStreamAPI.DocValidEdi.ToString());</code> |
| DocumentOnly This says that the EDI has no enveloping, or its enveloping should be ignored. It will be processed with the ISA and GS definitions specified in envelop. DocumentOnly and envelop must both be present if you want the enveloping to be processed this way. Another method for document-only processing is to use ignoreENV. | <code>insObj.flag = fsInStreamAPI.DocumentOnly;</code> |
| DS_DebugOptions 0 turns off debug Docsplitter messages 1 turns on debug Docsplitter messages | <code>docSp.DS_DebugOptions = 1;</code> |

| Methods | Example |
|--|--|
| <p>DS_inputFile</p> <p>When using InputByFile, this gives the full path and name of the EDI data file that will be validated.</p> <p>When using InputByMemory, this gives a name or title for the data to be output in the Start Message.</p> | <pre>docSp.DS_inputFile = insObj.inputFile; insObj.inputFile=INSTREAMROOT + @"DemoData\837I-Demo3.txt";</pre> |
| <p>DS_Options</p> <p>Docsplitter options to create valid and/or invalid EDI files using one of these:</p> <p>DS_ReportValidOnly</p> <p>DS_ReportInvalidOnly</p> <p>DS_ReportBoth</p> <p>DS_ReportInvalidFile</p> <p>Gives the path and filename of the Docsplitter invalid EDI file.</p> <p>DS_ReportValidFile</p> <p>Gives the path and filename of the Docsplitter valid EDI file.</p> | <pre>docSp.DS_Options = fsInStreamAPI.DS_ReportBoth; docSp.DS_ReportInvalidFile = INSTREAMROOT + @"Output\Doc_csharp_InValidFile.txt"; docSp.DS_ReportValidFile = INSTREAMROOT + @"Output\Doc_csharp_ValidFile.txt";</pre> |
| <p>DS_outputfile</p> <p>Gives the name of the validation detail results file for Docsplitter input.</p> | <pre>insObj.outputfile=INSTREAMROOT + @"Output\instreamcsharp_API_result1.TXT"; docSp.DS_outputfile = insObj.outputfile;</pre> |
| <p>DS_ReportFile</p> <p>Gives the path and filename of the Docsplitter report. It will contain XML if the filename ends with .xml. It will be a delimited report if the filename ends with .csv.</p> | <pre>instObj.DS_ReportFile=OUTPUTDIR+"My_Report.xml";</pre> |
| <p>DSUB_EDInputSourceOption</p> <p>Sets Docsplitter's EDI input to memory or file; used when Docsplitter is being used in the mode that calls Instream.</p> <p>Docsplitter can input from memory if it calls Instream, but not if it accepts validation detail results as input.</p> <p>This can be set to:</p> <p>InputByMemory</p> <p>InputByFile</p> | <pre>dsubObj.DSUB_EDInputSourceOption = fsInStreamAPI.InputByMemory;</pre> |

| Methods | Example |
|--|--|
| DSUB_guideline Points to Docsplitter's input guideline. This is needed when Docsplitter is being used in the mode that calls Instream. | <code>dsubObj.DSUB_guideline = "PDSA835";</code> |
| DSUB_EDIOutputFile Points to Docsplitter's EDI output file. | <code>dsubObj.DSUB_EDIOutputFile = INSTREAMROOT + @"Output\C#APIExample6_DataSwapper_EDI2.txt";</code> |
| DSUB_EDIOutputSourceOption Sets Docsplitter's report and EDI output to file. Currently, Docsplitter can output to file, but not to memory. This is set to: OutputbyFile | <code>dsubObj.DSUB_EDIOutputSourceOption = fsInStreamAPI.OutputByFile;</code> |
| DSUB_ReportFile Points to Docsplitter's report output file. | <code>dsubObj.DSUB_ReportFile = INSTREAMROOT + @"Output\C#APIExample6_DataSwapper_Report2.txt";</code> |
| DSUB_szEDIInputFile Points to Docsplitter's EDI output file. | <code>dsubObj.DSUB_szEDIInputFile = insObj.inputFile;</code> |
| DSUB_szInStreamFile Points to the Instream detail results file that is to be used for Docsplitter input. | <code>dsubObj.DSUB_szInStreamFile = insObj.outputfile;</code> |
| edibuffer Input parameter pointing to memory buffer where input EDI data is to be found. This parameter is required when InputByMemory is specified in the flag parameter (see below), and ignored otherwise. | <i>See example 2 in csharpDemo.cs.</i> |
| Edi_outputOption Specifies that the Docsplitter valid and invalid EDI output is to go to memory or file. Settings can be: OutputByFile OutputByMemory | <code>docSp.Edi_outputOption=fsInStreamAPI.OutputByMemory;</code> |

| Methods | Example |
|---|--|
| envelop Definition of the ISA and GS if DocumentOnly is used. | <pre>static string envelopISA="ISA*00* *00* *01*9012345720000 *01*9088877320000 *020108*1042*U*00200*000000001*0*T*!"; static string envGS="GS*HP*901234572000*908887732000*20020108*1 615*1*X*004010X091!"; insObj.envelop=envelopISA+"\n"+envGS; insObj.flag = fsInStreamAPI.DocumentOnly;</pre> |
| ErrorMessage Returns the text of the last program error encountered by Instream validation, Docsplitter, or Response Generator. | <pre>if(retVal != 100) { Console.WriteLine(fsInStreamAPI.ErrorMessage); }</pre> |
| fileInfo_mon fileInfo_day fileInfo_year fileInfo_hour fileInfo_min fileInfo_sec origFile_fsize OrigFilePathName Specifies the month, day, year, hour, minute, second, size, and path of the original EDI file. | <pre>insObj.fileInfo_mon=2; insObj.fileInfo_day=25; insObj.fileInfo_year=2005; insObj.fileInfo_hour=14; insObj.fileInfo_min=12; insObj.fileInfo_sec=34; insObj.origFile_fsize = 9999989; insObj.OrigFilePathName=@"C:\Documents and Settings\xxx\My Documents\abc.txt";</pre> |
| flag Describes whether the input and output is by memory or file. Set this flag before you run validation. InputByFile - The inputFile parameter contains a file name. OutputByFile - The outputFile parameter contains a file name. InputByMemory - (Instream validation only) the input EDI data is located in memory. The edibuffer parameter must also be set to point to the input buffer as described above. OutputByMemory - (Instream validation only) the detail and summary output are to go to memory. Use the Report routine (at the end of fsInStreamAPI.cs) to perform the actual update to memory. This routine must be customized. | <pre>insObj.flag= fsInStreamAPI.InputByMemory fsInStreamAPI.OutputByMemory;</pre> |

| Methods | Example |
|--|--|
| <p>guideline</p> <p>Gives the name of the guideline to be used for validation. If omitted, Instream uses the first guideline encountered that matches the data's Version/Release Identifier code (Segment GS Element 08). If you are using Docsplitter, be sure it is a GuidelinePlus or is based on a GuidelinePlus (one that starts with PD).</p> | <pre>insObj.guideline=@"PDSA837I";</pre> <p>or</p> <pre>insObj.guideline=@"MY850";</pre> |
| <p>inputFile</p> <p>When using InputByFile, this gives the full path and name of the EDI data file that will be validated.</p> <p>When using InputByMemory, this gives a name or title for the data to be output in the Start Message.</p> | <pre>insObj.inputFile=INSTREAMROOT + @"DemoData\837I-Demo3.txt";</pre> |
| <p>Instream DocSplit RespGen DataSubstitution</p> <p>Procedures that actually run validation, Docsplitter, Response Generator, and Docsplitter.</p> | <pre>int retVal=myObj.InStream(ref insObj); retVal=myObj.DocSplit(ref docSp) retVal=myObj.RespGen(ref respgen); retVal=apiObj.DataSubstitution(ref dsubObj);</pre> |
| <p>OutputBuf997 OutputBuf277 OutputBuf824 OutputBufTA1 OutputBufCus</p> <p>(Response Generator memory output only)</p> <p>These are the buffer names for the Response Generator output.</p> | <pre>sw.Write(InStreamAPI.fsInStreamAPI.OutputBuf997.ToString());</pre> |
| <p>outputFile</p> <p>Gives the name of the validation detail results file.</p> <p>When using OutputByFile, this specifies the full path name to the output detail file.</p> <p>When using OutputByMemory, this is a file name to be used if needed due to failure of memory output.</p> | <pre>insObj.outputfile=INSTREAMROOT +@"Output\instreamcsharp_API_result1.TXT";</pre> |

| Methods | Example |
|---|--|
| profile <p>Gives the name of the validation profile file.</p> <p>Default is \$fsdeflt.apf.</p> | insObj.Profile = "users.apf"; |
| Report <p>This writes EDI to memory.</p> <p>Please see the Report routine at the end of fsInStreamAPI.cs. You must override this routine to perform the actual output as desired.</p> <p>If you want to terminate validation, return the constant false.</p> <p>If you want to continue validation, return the constant true.</p> | |
| Report_FormatOption <p>(memory output only)</p> <p>Specifies the format of the Docsplitter report, one of these:</p> <p>DocXMLFormat DocCSVFormat</p> <p>The XML format slows performance more than the CSV format.</p> | docSp.Report_FormatOption = fsInStreamAPI.DocXMLFormat; |
| Report_outputOption <p>Specifies that the Docsplitter report is to be output to memory or file. Settings can be:</p> <p>OutputByFile OutputByMemory</p> | docSp.Report_outputOption=fsInStreamAPI.OutputByMemory; |
| RG_fs997Report RG_fs999Report RG_fs824Report RG_fs277Report RG_fsTA1Report RG_fsTextReport <p>(memory output only)</p> <p>If set to 1, these create the specific types of Response Generator output in memory.</p> | respngen.RG_fs824Report=1; respngen.RG_fsTA1Report=0; instObj.fs999Report = true; <p>Since RG_fs824Report is set to 1, this example creates 824 output in memory.</p> <p>It does not create TA1 output.</p> |

| Methods | Example |
|---|---|
| RG_Options Response Generator options, which can include any options mentioned in TIB_fsp-instream_<n.n>_respngen.pdf . | respngen.RG_Options="-ge -y"; <i>This example specifies that group enveloping is to be included in the response documents and that Response Generator can overwrite files if they exist.</i> Important: When specifying Response Generator options via the API, leave no space between the option and its value. For example, use -er3 instead of -er 3. Spaces may cause the API response generator call to fail. |
| RG_outputfile Gives the name of the validation detail results file for Response Generator input. | insObj.outputfile=INSTREAMROOT + @"Output\instreamcsharp_API_result1.TXT"; respngen.RG_outputfile = insObj.outputfile; |
| RG_Rep277 RG_Rep824 RG_Rep997 Gives the name of the EDI file(s) to be created by Response Generator. (For TA1 generation, use RG_Options with -gTA1 n.) | respngen.RG_Rep997=INSTREAMROOT + @"Output\RespGen_csharp_997.txt"; |
| respGenInputOption Specifies that the Instream detail results file that serves as input to Response Generator is in memory or in a file. It can be set to: InputByMemory InputByFile | respngen.RG_respGenInputOption = fsInStreamAPI.InputByMemory; |
| respGenOutputOption Specifies that the Response Generator output is to go to memory or to a file. It can be set to: OutputByMemory OutputbyFile | respngen.RG_respGenOutputOption = fsInStreamAPI.OutputByMemory; |
| RG_repText RG_Reptemp RG_repText gives the name of a custom text report to be created by Response Generator. RG_Reptemp gives the name of the template that controls the format of the custom text report. If you use one of these, use both. | respngen.RG_repText=INSTREAMROOT + @"Output\RespGen_csharp_text.txt"; respngen.RG_Reptemp=INSTREAMROOT + @"DemoData\RGtemplate837I_c.txt"; |

| Methods | Example |
|---|---|
| <p>seg_separator elm_separator comp_separator</p> <p>Separators for “document-only” validation - when the data does not have ISA or GS enveloping, or when the enveloping is to be ignored.</p> <p>They can be in these formats:</p> <p>integer example: 29</p> <p>hexadecimal example: 0x1E</p> <p>character example: ~</p> <p>ignoreENV</p> <p>Validation is to ignore ISA and GS enveloping and use the separators provided with seg_separator, elm_separator, and comp_separator.</p> <p>This requires that seg_separator, elm_separator, and comp_separator be provided, and it must come AFTER them.</p> <p>Another method for document-only processing is to use DocumentOnly (see page 38).</p> | <pre>insObj.seg_separator='~'; insObj.elm_separator='*'; insObj.comp_separator=':~'; insObj.ignoreENV=true;</pre> |
| <p>SummaryMessage</p> <p>Sends the summary report to memory when OutputByMemory is used.</p> | <pre>Console.WriteLine(fsInStreamAPI.summaryMessage);</pre> |
| <p>userMessage</p> <p>Specifies free-form text to be inserted in a GEN record with number 15078. Example:</p> <p>GEN 015078 1 0Sock 2</p> <p>This is for your own use. Transaction Insight can display it.</p> | <pre>insObj.userMessage="Sock 2";</pre> |

VB.NET API for Windows

Requirements

Please see the readme.txt file that accompanies Instream for Instream requirements information.

You will also need:

- Access to persons with knowledge of VB.NET and Instream.
- Microsoft Windows. There is no UNIX version of this API
- Microsoft Visual Studio 2008 or later.

Setting up your Environment

Put Instream's Bin directory in your path.

fsInStreamAPI

This uses the same methods as the C# API. See FSInStream Class on page [27](#).