Data Pipeline Primer

CTP Outline, draft

25-Sep-11

**Summary**

Amalgamate clinical decision-making information to be semantically interoperable and computationally sound.

# Data Pipeline Workflow

The data pipeline acquires feed data and transforms the data to a state such that entities can be bound to them. Feed data flows through the three configurable subsystems shown in the table that follows. The subsystems are interdependent; output from the Feed Acquisition subsystem delivers input to the Message Shredder subsystem and output from the Message Shredder subsystem delivers input to the Data Parser subsystem.

To learn about how to run a feed through the data pipeline, follow the vertical links of the Feed Acquisition, Message Shredder, and Data Parser subsystems.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Feed Acquisition** | **Message Shredder** | **Data Parser** |
| Summary | [snippet]  *more* | [snippet]  *more* | [snippet]  *more* |
| Input | MLLP  File  *more* | MSG201  *more* | Shredded columns  *more* |
| Output | MSG201  *more* | Shredded columns  *more* | Derived columns  *more* |
| Application  XML | FeedAcquisition.cab  *Create an instance* | shred.cab  *Create an instance* | parse.cab  *Create an instance* |
| Extensibility | Write an acquisition plug-in | Write a shredder plug-in | Write a custom transform  Write a parser plug-in  Write custom enterprise ID |
|  | Feed Acquisition (PDF) | Message Shredder (PDF) | Data Parser (PDF) |

# Prerequisites for building pipeline applications

* Full Amalga platform with all services (including DataPipelineStorage setup)
* Text editor

# Unpack CAB Files

**To locate and unpack all pipeline manifest XML files**

1. Create a staging folder with a nested temporary folder. For example, create C:\Pipeline Manifest to hold the cabinet files and C:\Pipeline Manifest\MasterConfig to hold the extracted XML files.
2. Navigate to xxx and locate FeedAcquisition.CAB, Shredder.CAB, and Parser.CAB**.**
3. Copy the set of cabinet files to the staging folder.
4. For each cabinet file, extract the XML files to the \MasterConfig subfolder. Keep the extracted XML files as master configuration files from you can generate application instances.
5. Create a subfolder for each new feed. For example, create C:\Pipeline Manifest\ADT for ADT feed.
6. Copy a full set of extracted XML configuration files to the feed folder. For example, copy the XML files from the master folder (C:\Pipeline Manifest \MasterConfig) to the feed folder (C:\Pipeline Manifest\ADT).

# Pipeline Performance Counters

The following table shows the performance counters that pertain to the pipeline subsystems.

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance counter name** | **Acquisition subsystem** | **Shredder subsystem** | **Parser subsystem** |
| Message Process Time | **Yes** | No | No |
| Number of Invalid Messages | **Yes** | No | No |
| Messages Received | No | **Yes** | **Yes** |
| Messages Per Second | **Yes** | **Yes** | **Yes** |
| Messages Successful | No | **Yes** | **Yes** |
| Messages Unsuccessful | No | **Yes** | **Yes** |
| AmalgaEnterpriseId Messages Total | No | No | **Yes** |
| AmalgaEnterpriseId Lookup Total | No | No | **Yes** |
| AmalgaEnterpriseId Deadlock Total | No | No | **Yes** |
| AmalgaEnterpriseId Matches EID | No | No | **Yes** |
| AmalgaEnterpriseId Matches GEID | No | No | **Yes** |
| AmalgaEnterpriseId Matches SOID | No | No | **Yes** |
| AmalgaEnterpriseId Matches OID | No | No | **Yes** |

# Feed Acquisition Subsystem

**Summary**

Move feed data from the source interface to the MSG201 table.

Transport to the Feed Acquisition subsystem can take place over the Internet, a wide-area network (WAN), or local-area network (LAN). All modes deliver high reliability, high throughput, and high availability.

You configure, publish, and deploy one acquisition application instance for each feed. For instructions, see Acquisition Application – Acquire a Feed.

**Receive and send adapters**

An application instance has one receive adapter (MLLP- or file-based) and one send adapter that identifies the destination message queue. In addition to specifying configuration settings to customize the application instance, you can write and publish your own application plug-in. For options, see Feed Acquisition Extension.

**MessageQueue and azQueue**

Side-by-side story

## Acquisition Input

Acquisition input is the format in which a source system submits messages to the pipeline. Your configuration varies depending on the input formats of the feed.

**Minimal Lower Layer Protocol (MLLP) to transport HL7 messages**

You must specify a feed name, feed type, and unique port number.

* You can configure secondary storage, such as a file system or database, as a backup storage to receive data when the connection to MSG201 is not available. The subsystem manages migrating temporarily stored data into MSG201 after you restore the connection. In addition, the Acquisition subsystem provides fail-over cluster (Active/Passive) mode for disaster recovery and failover. For details, see xxx.
* The receive adapter returns an acknowledgement (ACK) message unless the source system is unable or unwilling to receive ACK messages for the HL7 feed.

**File format to transport file-based messages, such as comma-separated value (CSV) format**

You must specify a feed name, feed type, and the source directory (and search pattern) of the file.

* You can configure an alternative directory for large messages files sent from the source system. Large messages are typically greater than 2 GB.

## Acquisition Output

Valid output appears in the MSG201 database table of the message queue specified in the acquisition service manifest. Valid output is ready for message shredding.

**Rejected messages**

The acquisition subsystem stores rejected HL7 messages in MSG201, or secondary storage, with the original message type of the feed. Both accepted and rejected messages remain together, thereby ensuring the proper ordering of messages. Your parsers are responsible for determining whether to skip the rejected message or stop the parser from further processing.

## Acquisition Application – Acquire a Feed

Use the procedures that follow to configure, publish, and deploy an instance of the acquisition application for the specified feed.

Note:   
When you complete the procedures in this topic, feed data will begin to accumulate in the MessageQueue database.

**Procedures in this topic**

* **Configure acquisition instance names**
* **Configure the service for file-based acquisition (CSV)**
* **Configure the service for MLLP acquisition (HL7)**
* **Publish an application instance**
* **Deploy an application instance**

### ****C**onfigure acquisition instance names**

Establish one application instance name and one service instance name for each feed.

**To configure acquisition instance names**

1. From the feed folder, locate the application manifest (AcquisitionApplication.XML). To obtain pipeline manifest masters, see Unpack CAB Files.
   1. Open the file with Notepad, or an equivalent text editor, and save it using the feed name. For example, save as AcquisitionApplication\_ADT.XML to create an application instance for the ADT feed.
   2. Edit the InstanceName element value in the file. Save and then close the file. For a configuration example, see Acquisition Application Manifest.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | The instance name and the file name must match each other. Appending the name of the feed to the instance name can help you identify the application.  **Application instance examples**   * AcquisitionApplication\_ADT * AcquisitionApplication\_CsvFile1   **Service instance examples (Step 2)**   * AcquisitionService\_ADT * AcquisitionService\_CsvFile1 |

1. Repeat Step 1 to edit the service instance name. Name the instance as described in Step 1 and then save the file using the feed name (for example, AcquisitionService\_ADT and AcquisitionService\_ADT.XML).
2. Without closing the acquisition service manifest, go to the next procedure (choose one):
   * **To configure the service for file-based acquisition (CSV)**
   * **To configure the service for MLLP acquisition (HL7)**

### ****C**onfigure the service for file-based acquisition (CSV)**

Use following configuration recommendations for feeds that contain comma-separated value (CSV) data. For a description, see Acquisition Input.

**To configure the service for file-based acquisition**

For a configuration example, see File Service Manifest.

1. In the acquisition service manifest, locate and modify the configuration settings shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute or  child element** | **Unique** | **Description** |
| <Feeds>  <Feed> | name | Yes | The application appends the feed name you specify to the path of the folders it generates.  **Example**  CsvFile\_1 |
| <Feeds>  <Feed>  <receiveadapter> | <directory> | No | Destination folder for incoming files sent from the source system. The application needs access permission to the folder.  **Example**  L:\ Receive\File |
| <Feeds>  <Feed> | <msgtype> | No | Type of message submitted to the application by the source system.  **Examples CSV** |
| <Feeds>  <Feed> | <largemsgdirectory> | No | Alternative directory for large messages files sent from the source system.  **Example**  L:\LargeMessageDirectory |

1. Close and save the file.
2. Go to the next procedure: **To publish an application instance.**

### ****Configure the service for MLLP acquisition (HL7)****

Use the following configuration recommendations for feeds that comply with HL7 standards. For a description, see Acquisition Input.

**To configure the service for MLLP acquisition**

For a configuration example, see MLLP Service Manifest.

1. In the acquisition service manifest, locate and modify the configuration settings shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute or  child element** | **Unique** | **Description** |
| <Feeds>  <Feed> | name | Yes | The application appends the feed name you specify to the path of the folders it generates.  **Example**  ADT |
| <Feeds>  <Feed>  <receiveadapter> | <port> | Yes | Active port for an MLLP feed. **Example**  20000 |
| <Feeds>  <Feed>  <tempstorageadapter> | <directory> | No | Fallback directory if SQL Server is unavailable. The application writes message files to this directory and then submits them to the MessageQueue database when SQL Server becomes available.  **Example**  L:\Temp\SecondaryStorage |
| <Feeds>  <Feed> | <msgtype> | No | Type of message submitted to the application by the source system.  **Examples**  ADT |

1. Close and save the file.
2. Go to the next procedure: **To publish an application instance.**

### ****Publish an application instance****

Perform the following sequence from ???

**To publish an application instance**

1. Run Windows PowerShell as administrator and change directory using the following command:

**cd C:\Program Files\Microsoft Amalga\System 3.0\Platform\PowerShell**

1. Import the Amalga Hosted Runtime library (one time).

**import-module .\Microsoft.Amalga.Deployment.RuntimeLib.dll  
import-module .\Microsoft.Amalga.Deployment.TopologyLib.dll**

1. Publish an instance of the application using the following syntax:

**Publish-ApplicationInstance -ApplicationCabFile** *FullPathOfApplication***.cab -ApplicationAuxiliaryFiles "***FullPathOfAppManifest***.xml;***FullPathOfServiceManifest***.xml"**

For example, type the following command to publish an instance of the acquisition application using the specified CAB and auxiliary files:

**Publish-ApplicationInstance -ApplicationCabFile C:\Manifest\Acquisition.cab   
-ApplicationAuxiliaryFiles "C:\Manifest\AcquisitionApplication.xml;C:\Manifest\AcquisitionService.xml"**

### ****Deploy an application instance****

Perform the following sequence from ???

**To deploy an application instance**

1. Type the following command to deploy the application instance to the physical host:

**Sync-PhysicalHost**

1. From **Start**, run **services.msc** to open the Services MMC. Verify that your service appears in the service list.

Right-click the service and point to **Start**.

## Feed Acquisition Extension

To be supplied.

# Message Shredder Subsystem

**Summary**

Split each message into the tables and columns based on the structure of the message or file.

The Message Shredder subsystem automatically reads message data from the specified message queue, analyzes the messages, creates a SQL Server schema to hold each data field, and inserts the data into the schema. An individual shredder (one instance of the Message Shredder application) works with data from a single feed. The subsystem shreds all feeds in parallel.

Message structure, not message content, determines the storage schema for each message type. For details, see Message Shredder Output.

While the configuration requirements are minimal for message shredder instances in this subsystem, you have useful options to increase the odds that users see the right information. The following options typically apply to complex feeds:

* Derived columns

Derived columns are special columns that store the data required to derive a primary context key used for HL7 incremental collapsing. You can apply an optional filter if you must extract the derived column from a repeating field (xml column).

* Context key columns

A context key uniquely identifies an entity and can be a concatenation of values that exist strictly within a feed. You can define multiple context keys to identify multiple entities. An individual context key can have multiple levels that describe the same entity by alternate sets of data (MRN vs. social security number and first name).

* Parent-child relationships

Some elements, particularly in the HL7 messages, are hierarchical in nature and you must configure the shredder service manifest to describe the relationships. Relationship information allows the data that a shredder inserts into separate tables to rejoin later.

## Message Shredder Input

During every polling interval, a message shredder instance reads a batch of raw messages from the specified message queue. You can configure the number of messages (or rows in a file) in the batch for each message type. The default batch size is xxx.

**Comma-separated value format**  
*item1*, *item2*, *item3*, *item4*  
*item5*, *item6*, *item7*, *itme8*

**HL7 version 2.5 format**  
MSH|^~\&|GHH LAB|ELAB-3|GHH OE|BLDG4|200202150930||ORU^R01|CNTRL-3456|P|2.4

PID|||555-44-4444||EVERYWOMAN^EVE^E^^^^L|JONES|19620320|F|||153 FERNWOOD DR.^

^STATESVILLE^OH^35292||(206)3345232|(206)752-121||||AC555444444||67-A4335^OH^20030520

OBR|1|845439^GHH OE|1045813^GHH LAB|15545^GLUCOSE|||200202150730|||||||||

555-55-5555^PRIMARY^PATRICIA P^^^^MD^^|||||||||F||||||444-44-4444^HIPPOCRATES^HOWARD H^^^^MD

OBX|1|SN|1554-5^GLUCOSE^POST 12H CFST:MCNC:PT:SER/PLAS:QN||^182|mg/dl|70\_105|H|||F

OBX|2|SN|1554-6^GLUCOSE^POST 24H CFST:MCNC:PT:SER/PLAS:QN||^182|mg/dl|70\_105|H|||F

## Message Shredder Output

Database schema of tables and views in the following canonical form:

* Tables  
  [*subsystem*].[*feedname*].[*segmentname*]
* Views  
  [*subsystem*].[*feedname*].[V\_COLLAPSE\_*segmentname*]

The shredded columns are a mechanical representation of an incoming feed data in which each field generally yields a column.

Values in the shredded columns will always be the raw input values with no transformations performed on them. For example, an HL7 element represented as || will be stored as an empty string, and |””| will be stored as “”.

## Message Shredder Application – Shred Feed Data

**Summary**

Configure and deploy an instance of the Message Shredder application to generate the first generation of tables and columns for the specified feed.

Use the procedures that follow to edit the manifest, publish the cabinet file, and deploy an instance of the application. For all manifest templates that relate to pipeline applications, see Unpack Manifest Files.

Note:   
When you complete the procedures in this topic, feed data will begin to accumulate in the ClinicalData database.

**Procedures in this topic**

* **Configure shredder instance names**
* **Configure the service for file-based acquisition (CSV)**
* **Configure the service for MLLP acquisition (HL7)**
* **Publish an application instance**
* **Deploy an application instance**

### ****C**onfigure shredder instance names**

Establish one application instance name and one service instance name for each feed.

**To configure shredder instance names**

1. From the feed folder, locate the application manifest (ShredderApplication.XML). To obtain pipeline manifest masters, see Unpack CAB Files.
   1. Open the file with Notepad, or an equivalent text editor, and save it using the feed name. For example, save as AcquisitionApplication\_ADT.XML to create an application instance for the ADT feed.
   2. Edit the InstanceName element value in the file. Save and then close the file. For a configuration example, see Acquisition Application Manifest.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | The instance name and the file name must match each other. Appending the name of the feed to the instance name can help you identify the application.  **Application instance examples**   * ShredderApplication\_ADT * ShredderApplication\_CsvFile1   **Service instance examples (Step 2)**   * ShredderService\_ADT * ShredderService\_CsvFile1 |

1. Repeat Step 1 to edit the service instance name. Name the instance as described in Step 1 and then save the file using the feed name (for example, AcquisitionService\_ADT and AcquisitionService\_ADT.XML).
2. Without closing the acquisition service manifest, go to the next procedure (choose one):
   * **To configure the service for file-based acquisition (CSV)**

**To configure a Shredder instance**

1. Locate the application manifest template (ShredderApplication.XML), edit the values shown in the table that follows, and then save the file. For a configuration example, see Shredder Application Manifest.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | Appending the name of the feed to the instance name can help you identify the application.  **Example**  ShredderApplication\_ADT |

1. Locate the server manifest template (ShredderService.XML), edit the values shown in the table that follows, and then save the file. For a configuration example, see Shredder Service Manifest.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute and/or  child element** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | Appending the name of the feed to the instance name can help you identify the service.  **Example**  ShredderService\_ADT |
| <Settings>  <Setting =name> | L1DatabaseConnectionString  <Default> | No | Default string used to connect to the Message Queue. |
| <Settings>  <Setting =name> | G1DatabaseConnectionString  <Default> | No | Default string used to connect to the ClinicalData database. This is where the results of the message shredder will go. |
| <Settings>  <Setting =name> | DatabaseSchema | No |  |
| <Settings>  <Setting =name> | FeedName | No |  |
| <Settings>  <Setting =name> | LastProcessedSequenceNumber |  | Always start at zero. |
| <Settings>  <Setting =name> | BatchSize |  |  |
| <Settings>  <Setting =name> | ShredderPluginConfigurationXml  <DefaultValue>  <Plugin>  <URL> | No | The correct path must correspond to the InstanceName, as it is used in creating the path to the assemblies. |

**To publish the application instance**

1. Run Windows PowerShell as administrator and change directory using the following command:

**cd C:\Program Files\Microsoft Amalga\System 3.0\Platform\PowerShell**

1. Import the Amalga Hosted Runtime library (one time).

**import-module .\Microsoft.Amalga.Deployment.RuntimeLib.dll  
import-module .\Microsoft.Amalga.Deployment.TopologyLib.dll**

1. Publish an instance of the application using the following syntax:

**Publish-ApplicationInstance -ApplicationCabFile** *FullPathOfApplication***.cab -ApplicationAuxiliaryFiles "***FullPathOfAppManifest***.xml;***FullPathOfServiceManifest****.xml*"**

For example, type the following command to publish an instance of the Message Shredder application using the specified CAB and auxillary files:

**Publish-ApplicationInstance -ApplicationCabFile C:\Manifest\FirstGenerationParser.cab -ApplicationAuxiliaryFiles "C:\Manifest\FirstGenerationParserApplication.xml;C:\Manifest\FirstGenerationParserService.xml"**

**To deploy the application instance**

1. Type the following command to deploy the application instance to the physical host:

**Sync-PhysicalHost**

1. From **Start**, run **services.msc** to open the Services MMC. Verify that your service appears in the service list.
2. Right-click the service and point to **Start**.

## Message Shredder Extension

To be supplied.

# Data Parsing

To be supplied.

## Data Parser Input

To be supplied.

## Data Parser Output

To be supplied.

## Data Parser Tagging

To be supplied.

## Data Parser Application

**Summary**

Configure and deploy an instance of the Data Parser application to generate the second generation of tables and columns for the specified feed.

Use the procedures that follow to edit the manifest, publish the cabinet file, and deploy an instance of the application. For all manifest templates that relate to pipeline applications, see Unpack Manifest Files.

**To configure a Parser instance**

1. Locate the application manifest template (ParserApplication.XML), edit the values shown in the table that follows, and then save the file. For a configuration example, see Parser Application Manifest.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | Append the message type to ParserApplication. For example, ParserApplication\_ADT |

1. Locate the service manifest template (ParserService.XML). For a configuration example, see Parser Service Manifest.
2. Edit the manifest to specify the following service settings:

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute and/or  child element** | **Unique** | **Description** |
| <DeploymentManifest> | InstanceName | Yes | Append the message type to ParserService. For example, ParserService\_ADT |
| <Settings>  <Setting =name> | ClinicalDatabaseConnectionString  <Default> | No | Default string used to connect to the ClinicalDatabase. |
| <Settings>  <Setting =name> | ClinicalDatabaseAdminConnectionString  <Default> | No | Default string used to connect to the ClinicalData database. |
| <Settings>  <Setting =name> | DatabaseSchema | No |  |
| <Settings>  <Setting =name> | FeedName | No |  |
| <Settings>  <Setting =name> | LastProcessedSequenceNumber |  | Always start at zero. |
| <Settings>  <Setting =name> | BatchSize |  |  |
| <Settings>  <Setting =name> | FeedConfigurationXML |  | CDATA with several sections of configuration. |
| <DataParserConfiguration>  <MessageShredderInput> | Segment |  | The segments (tables) that you wish to pull fields from. |
| <DataParserConfiguration>  <DataParserOutput> | Segment  Element |  | The segments (tables) you wish to output fields to, as well as the Elements (fields) you wish to output |
| <DataParserOutput>  <Elements>  <Element> | Transform |  | The action you wish to take on the data you output. This is a data transformation step. |
| <ExtensibleDataParserOutput> | Segment |  |  |
| <DataConstants> |  |  |  |
| <Plugins> | Plugin |  | This contains our plug-ins we ship with the product, as well as customer defined plugins (such as a custom transform library) that can be loaded and used. |
| <EnterpriseId>  <InputIdentifiers> | SourceIdentifier |  |  |
| <Settings>  <Setting =name> | ShredderPluginConfigurationXml  <DefaultValue>  <Plugin>  <URL> | No | The correct path must correspond to the InstanceName, as it is used in created the path to the assemblies. |

**To publish the application instance**

1. Run Windows PowerShell as administrator and change directory using the following command:

**cd C:\Program Files\Microsoft Amalga\System 3.0\Platform\PowerShell**

1. Import the Amalga Hosted Runtime library (one time).

**import-module .\Microsoft.Health.Deployment.RuntimeLib.dll**

1. Publish an instance of the application using the following syntax:

**Publish-ApplicationInstance -ApplicationCabFile** *FullPathOfApplication***.cab -ApplicationAuxiliaryFiles "***FullPathOfAppManifest***.xml;***FullPathOfServiceManifest****.xml*"**

For example, type the following command to publish an instance of the Data Parser application using the specified CAB and auxillary files:

**Publish-ApplicationInstance -ApplicationCabFile C:\Manifest\SecondGenerationParser.cab -ApplicationAuxiliaryFiles "C:\Manifest\SecondGenerationParserApplication.xml;C:\Manifest\SecondGenerationParserService.xml"**

**To deploy the application instance**

1. Type the following command to deploy the application instance to the physical host:

**Sync-Components**

## Data Parser Extension

To be supplied.

# Data Pipeline Configuration Samples

To be supplied.

# Feed Acquisition: Application Manifest Example

Modify each value marked with gray text highlighting as needed. For value descriptions, see Acquisition Application – Acquire a Feed.

<?xml version="1.0" encoding="utf-8"?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "Application"

Name="AcquisitionApplication"

InstanceName="AcquisitionApplication-Mllp\_0-129560894408230995"

Id ="17dd8829-5b50-46d0-b3a6-4dcda4f28742"

Version ="1.0.0.0">

<Settings>

</Settings>

<Dependencies>

</Dependencies>

<ApplicationManifest Publisher ="Microsoft" AmalgaMinVersion ="1.0.0.0" />

</DeploymentManifest>

# Feed Acquisition: Service Manifiest Example for MLLP Format

Modify each value marked with gray text highlighting as needed. For value descriptions, see Acquisition Application – Acquire a Feed.

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

Type = "WindowsWorker"

Name="AcquisitionService"

Id="ff003bed-0e89-48ea-af49-51bf20621729"

InstanceName="AcquisitionService\_ADT1"

Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName = "FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="Microsoft.Health.Acquisition.Core.exe"/>

<PerformanceCategories>

<PerformanceCategory Name="Amalga Data Acquisition Service Stats" Version="" CodeName="AcquisitionService" Help="Store the stats for Data Acquisition" MultiInstance="Yes">

<PerformanceCounter Name="Messages Per Second" CodeName="MessagesProcessedPerSec" Type="RateOfCountsPerSecond32" Help="Records the service activity rate as the number of messages processed per second" />

<PerformanceCounter Name="Message Process Time" CodeName="LastMessageProcessTime" Type="NumberOfItems64" Help="Records the time spent to process the last message" />

<PerformanceCounter Name="Number of Invalid Messages" CodeName="InvalidMessageProcessedCount" Type="NumberOfItems64" Help="Records the number of invalid messages processed" />

</PerformanceCategory>

</PerformanceCategories>

<Settings>

<Setting Name="MessageQueueConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=MessageQueue;uid=az-sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="SecondaryStorageFolderFullName" Type="System.String">

<DefaultValue>temp</DefaultValue>

</Setting>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[<?xml version="1.0" encoding="utf-8"?>

<feeds>

<feed name="ADT1">

<receiveadapter name="Microsoft.Health.Acquisition.Adapters.MLLPServer">

<port>20000</port>

<startblockcharacter>0x0B</startblockcharacter>

<endblockcharacter>0x1C</endblockcharacter>

<carriagereturncharacter>0x0D</carriagereturncharacter>

<ackcapture>(?&lt;tok0&gt;[^|]\*)\|(?&lt;tok1&gt;[^|]\*)\|(?&lt;tok2&gt;[^|]\*)\|(?&lt;tok3&gt;[^|]\*)\|(?&lt;tok4&gt;[^|]\*)\|(?&lt;tok5&gt;[^|]\*)\|(?&lt;tok6&gt;[^|]\*)\|(?&lt;tok7&gt;[^|]\*)\|(?&lt;tok8&gt;[^|]\*)\|(?&lt;tok9&gt;[^|]\*)\|(?&lt;tok10&gt;[^|]\*)\|(?&lt;tok11&gt;[^|]\*)\|(?&lt;tok12&gt;[^|]\*)\|(?&lt;tok13&gt;[^|]\*)</ackcapture>

<ackformular>MSH|&lt;tok1&gt;|&lt;tok4&gt;|&lt;tok5&gt;|&lt;tok2&gt;|&lt;tok3&gt;|$date||ACK^&lt;tok8&gt;|&lt;tok9&gt;|&lt;tok10&gt;|2.1|&lt;tok12&gt;\rMSA|AA|&lt;tok9&gt;||&lt;tok12&gt;</ackformular>

</receiveadapter>

<sendadapter name="Microsoft.Health.Acquisition.Adapters.SqlStorage">

<connectionstring>Data Source=localhost;Initial Catalog=MessageQueue;Integrated Security=True</connectionstring>

</sendadapter>

<tempstorageadapter name="Microsoft.Health.Acquisition.Adapters.FileStorage">

<directory>E:\SecondaryStorage</directory>

</tempstorageadapter>

<msgtype>ADT0</msgtype>

<KeepOnTemporaryStorage>false</KeepOnTemporaryStorage>

</feed>

</feeds>]]>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

# Feed Acquisition: Service Manifest Example for File Format

Modify each value marked with gray text highlighting. For value descriptions, see Acquisition Application – Acquire a Feed.

//File example

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "WindowsWorker"

Name="AcquisitionService"

InstanceName="AcquisitionService\_File1"

Id="ff003bed-0e89-48ea-af49-51bf20621729"

i:Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName="FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="Microsoft.Health.Acquisition.Core.exe" />

<Settings>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[<feeds>

<feed name="File\_1">

<receiveadapter name="Microsoft.Health.Acquisition.Adapters.FileAdapter" assembly="Microsoft.Health.Acquisition.Adapters, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null">

<directory>E:\Receive\File</directory>

<searchpattern>\*.\*</searchpattern>

</receiveadapter>

<sendadapter name="Microsoft.Health.Acquisition.Adapters.SqlStorage" assembly="Microsoft.Health.Acquisition.Adapters, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null">

<connectionstring>server=localhost;database=MessageQueue;Integrated security=SSPI</connectionstring>

</sendadapter>

<guid>5610cf61-fe97-42b1-9f4a-ee158362432d</guid>

<msgtype>ADT.1</msgtype>

<largemsgdirectory>E:\LargeMessageDirectory</largemsgdirectory>

</feed>

</feeds>]]>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

# Message Shredder: Application Manifest Example

Modify each value marked with gray text highlighting as needed. For value descriptions, see xxx.

<?xml version="1.0" encoding="utf-8"?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "Application"

Name="FirstGenerationParserApplication"

InstanceName="FirstGenerationParserApplication\_Ins1"

Id ="9cd7c3b1-2a45-4e14-844d-2629d597f554"

Version ="1.0.0.0">

<Settings>

<!--

<Setting Name="SampleSetting" Type="System.String" >

<DefaultValue>

</DefaultValue>

</Setting>

-->

</Settings>

<Dependencies>

<Dependency Name="TerminologyAppDependency" InstanceName="TerminologyApplication\_Ins1" Type="Application" Id="c031033a-49be-4c8f-a6a8-cb95bddaac0a" MinVersion="1.0.0.0" />

</Dependencies>

<ApplicationManifest Publisher ="Microsoft" AmalgaMinVersion ="1.0.0.0" />

</DeploymentManifest>

# Message Shredder: Service Manifest Example

Modify each value marked with gray text highlighting as needed. For value descriptions, see xxx.

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "WindowsWorker"

Name="FirstGenerationParserService"

InstanceName="G1ParserInstance\_CLAB\_ORU"

Id="2684661F-AAE0-40C1-9AC9-2D8061CEA8C6"

i:Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName="FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="FirstGenerationParser.exe" />

<PerformanceCategories>

<PerformanceCategory Name="Amalga Message Shredder Service Stats" Version="" CodeName="MessageShredder" Help="Store the stats for the Message Shredder" MultiInstance="Yes">

<PerformanceCounter Name="Messages Received" CodeName="MessagesReceived" Type="NumberOfItems64" Help="Records the the total number of messages received by the Message Shredder" />

<PerformanceCounter Name="Messages Per Second" CodeName="MessagesProcessedPerSec" Type="RateOfCountsPerSecond32" Help="Records the number of messages received per second" />

<PerformanceCounter Name="Messages Successful" CodeName="MessagesSuccessful" Type="NumberOfItems64" Help="Records the total number of messages processed successfully" />

<PerformanceCounter Name="Messages Unsuccessful" CodeName="MessagesUnsuccessful" Type="NumberOfItems64" Help="Records the number of messages that failed during processing" />

</PerformanceCategory>

</PerformanceCategories>

<Settings>

<Setting Name="L1DatabaseConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=MessageQueue;uid=az-sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="G1DatabaseConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!;Connection Timeout=60</DefaultValue>

</Setting>

<Setting Name="G1DatabaseName" Type="System.String">

<DefaultValue>ClinicalData</DefaultValue>

</Setting>

<Setting Name="DatabaseSchema" Type="System.String">

<DefaultValue>CLAB\_ORU</DefaultValue>

</Setting>

<Setting Name="FeedName" Type="System.String">

<DefaultValue>CLAB.ORU</DefaultValue>

</Setting>

<Setting Name="LastProcessedSequenceNumber" Type="System.Int32">

<DefaultValue>0</DefaultValue>

</Setting>

<Setting Name="BatchSize" Type="System.Int32">

<DefaultValue>500</DefaultValue>

</Setting>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[

<?xml version="1.0" encoding="utf-8"?>

<FirstGenerationConfiguration>

<Segments>

<Segment Name="PID" SourceName="PID" Description="Personal Identification">

<DerivedElements>

<DerivedElement Name="MRN" Description="Patient MRN" SemanticLabel="PID-3.Repeating" Filter="/Repetitions/Repetition[2]/PID-3.1" />

</DerivedElements>

<Entities>

<Entity Name="Patient">

<ContextKeys>

<Key Segment="PID" SemanticLabel="MRN" Name="MRN" Description="MRN from the second repetition" />

</ContextKeys>

</Entity>

<Entity Name="Account">

<ContextKeys>

<Key Segment="PID" SemanticLabel="PID-18.1" Name="Account" Description="Account" />

</ContextKeys>

</Entity>

</Entities>

</Segment>

<Segment Name="PV1" SourceName="PV1" ParentName="PID" Description="Patient Visit">

<DerivedElements />

<Entities>

<Entity Name="Visit">

<ContextKeys>

<Key Segment="PV1" SemanticLabel="PV1-19" Name="Visit" Description="Patient visit" />

</ContextKeys>

</Entity>

</Entities>

</Segment>

<Segment Name="OBR" SourceName="OBR" ParentName="PV1" Description="Observation Request">

<DerivedElements />

<Entities>

<Entity Name="LabRequest">

<ContextKeys>

<Key Segment="OBR" SemanticLabel="OBR-2" Name="PlacerOrderNumber" Description="Placer order number." />

</ContextKeys>

</Entity>

</Entities>

</Segment>

<Segment Name="OBX" SourceName="OBX" ParentName="OBR" Description="Observation Result">

<DerivedElements />

<Entities>

<Entity Name="LabResult">

<ContextKeys>

<Key Segment="OBR" SemanticLabel="OBR-4" Name="UniversalServiceId" Description="UniversalServiceId is defined in parent (OBR)." />

</ContextKeys>

</Entity>

</Entities>

</Segment>

<Segment Name="OBR\_NTE" SourceName="NTE" ParentName="OBR" Description="Notes and comments">

<DerivedElements />

<Entities />

</Segment>

<Segment Name="OBX\_NTE" SourceName="NTE" ParentName="OBX" Description="Notes and comments">

<DerivedElements />

<Entities />

</Segment>

<Segment Name="MSH" SourceName="MSH">

<DerivedElements />

<Entities />

</Segment>

</Segments>

</FirstGenerationConfiguration>

]]>

</DefaultValue>

</Setting>

<Setting Name="ShredderPluginConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[<Plugin Name="MessageHelper" Assembly="Microsoft.Health.Pipeline.FirstGenerationShredders, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396" ClassName ="HL7Shredder">

<NameSpace>Microsoft.Health.Pipeline.FirstGenerationShredders</NameSpace>

<URL>C:\Program Files\Microsoft Amalga\System 3.0\Application Worker Services\G1ParserInstance\_CLAB\_ORU\Microsoft.Health.Pipeline.FirstGenerationShredders.dll</URL>

<SourceType>FileSystem</SourceType>

</Plugin>]]>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

# Data Parser: SecondGeneration Application Manifest Example

Modify each value marked with gray text highlighting as needed. For value descriptions, see xxx.

<?xml version="1.0" encoding="utf-8"?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "Application"

Name="SecondGenerationApplication"

InstanceName="DataParserApplication\_ADT\_G"

Id ="576d0ffe-c339-4b15-852f-58d66315b736"

Version ="1.0.0.0">

<Settings>

<!--

<Setting Name="SampleSetting" Type="System.String" >

<DefaultValue>

</DefaultValue>

</Setting>

-->

</Settings>

<Dependencies>

<!-- Dependency Name="SomeDependencyApplication" Type="ApplicationDependency" Id="1d12ded1-c1f6-4d7e-9bdf-c687d67e0a45" MinVersion="1.0.0.0" /-->

</Dependencies>

<ApplicationManifest Publisher ="Microsoft" AmalgaMinVersion ="1.0.0.0" />

</DeploymentManifest>

# Data Parser: SecondGeneration Service Manifest Example

Modify each value marked with gray text highlighting as needed. For value descriptions, see xxx.

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

Type = "WindowsWorker"

Name="SecondGenerationService"

InstanceName="DataParserService\_ADT\_G"

Id="44162965-1f39-4f81-94b4-63002da90b90"

Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName="FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="Microsoft.Health.Pipeline.SecondGeneration.exe"/>

<PerformanceCategories>

<PerformanceCategory Name="Amalga Data Parser Service Stats" Version="" CodeName="DataParser" Help="Store the stats for the Data Parser" MultiInstance="Yes">

<PerformanceCounter Name="Messages Received" CodeName="MessagesReceived" Type="NumberOfItems64" Help="Records the the total number of messages received by the Data Parser" />

<PerformanceCounter Name="Messages Per Second" CodeName="MessagesReceivedSec" Type="RateOfCountsPerSecond32" Help="Records the number of messages received per second" />

<PerformanceCounter Name="Messages Successful" CodeName="MessagesProcessedSuccessful" Type="NumberOfItems64" Help="Records the total number of messages processed successfully" />

<PerformanceCounter Name="Messages Unsuccessful" CodeName="MessagesProcessedUnSuccessful" Type="NumberOfItems64" Help="Records the number of messages that failed during processing" />

</PerformanceCategory>

<PerformanceCategory Name="Amalga Enterprise ID" Version="" CodeName="AmalgaIdPerfCounters" Help="Amalga Enterprise Id Processor Counters" MultiInstance="Yes">

<PerformanceCounter Name="AmalgaEnterpriseId Messages Total" CodeName="MessagesTotal" Type="numberOfItems64" Help="The total # of messages run through AmalgaID"/>

<PerformanceCounter Name="AmalgaEnterpriseId Lookup Total" CodeName="LookupTotal" Type="numberOfItems64" Help="The total # of matching lookups performed"/>

<PerformanceCounter Name="AmalgaEnterpriseId Deadlock Total" CodeName="DeadlockTotal" Type="numberOfItems64" Help="The total # of submissions that were SQL deadlocked"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches EID" CodeName="MatchesEID" Type="numberOfItems64" Help="The # of lookups that result in an EID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches GEID" CodeName="MatchesGEID" Type="numberOfItems64" Help="The # of lookups that result in an GEID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches SOID" CodeName="MatchesSOID" Type="numberOfItems64" Help="The # of lookups that result in an SOID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches OID" CodeName="MatchesOID" Type="numberOfItems64" Help="The # of lookups that result in an OID match"/>

</PerformanceCategory>

</PerformanceCategories>

<Settings>

<Setting Name="ClinicalDatabaseConnectionString" Type="System.String">

<!--TODO: change this back to az-sa, and grant appropriate authorization-->

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="ClinicalDatabaseAdminConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="DatabaseSchema" Type="System.String">

<DefaultValue>ADT\_G</DefaultValue>

</Setting>

<Setting Name="FeedName" Type="System.String">

<DefaultValue>ADT.G</DefaultValue>

</Setting>

<Setting Name="LastProcessedSequenceNumber" Type="System.Int32">

<DefaultValue>0</DefaultValue>

</Setting>

<Setting Name="BatchSize" Type="System.Int32">

<DefaultValue>500</DefaultValue>

</Setting>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[

<?xml version="1.0" encoding="utf-8" ?>

<DataParserConfiguration>

<!--Section I, user defines the input message structure. -->

<MessageShredderInput>

<Segment Name="PID" SourceName="PID">

<Elements>

<Element Name="PID\_5\_1" />

<Element Name="PID\_5\_2" />

</Elements>

</Segment>

</MessageShredderInput>

<!-- Section II, user defines the output message structure-->

<DataParserOutput>

<Segment Name="PID" SourceName="PID">

<Elements>

<Element Name="FName" Type="System.String" MaxLength="50">

<ConceptGuid>11627229-A030-41C4-B323-9C5B8DAE0384</ConceptGuid>

<Transform>

<Action>ToTitleCase</Action>

<InputParameters>

<Parameter Name="LowCaseString" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_5\_1" />

</InputParameters>

</Transform>

</Element>

<Element Name="LName" Type="System.String" MaxLength="50">

<ConceptGuid>762F8617-40E0-4590-85F0-589AA98D6016</ConceptGuid>

<Transform>

<Action>ToTitleCase</Action>

<InputParameters>

<Parameter Name="LowCaseString" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_5\_2" />

</InputParameters>

</Transform>

</Element>

<Element Name="FullName" Type="System.String" MaxLength="150">

<ConceptGuid>4CF03880-FCA7-49DE-A396-345B480E7441</ConceptGuid>

<Transform>

<Action>Concatenate</Action>

<InputParameters>

<Parameter Name="Separator" SourceType="Constant" SourceName="Space" />

<Parameter Name="FirstName" SourceType="DerivedField" SourceTable="PID" SourceName="FName" />

<Parameter Name="LastName" SourceType="DerivedField" SourceTable="PID" SourceName="LName" />

</InputParameters>

</Transform>

</Element>

</Elements>

</Segment>

</DataParserOutput>

<ExtensibleDataParserOutput>

<Segment Name="PID" SourceName="PID">

<ConceptGuid>16452154-0BE5-47BD-A527-28A244D3EDD0</ConceptGuid>

<Elements>

<Element Name="Gender" Type="System.String" MaxLength="50" />

</Elements>

</Segment>

</ExtensibleDataParserOutput>

<!--Section III, user defines the constants, which will be used in G2 parsing.-->

<DataConstants>

<Element Name="Space" Type="System.String" Value=" " />

<Element Name="IndependenceDay" Type="System.DateTime" Value="07/07/2011" />

<Element Name="OBRChildTable" Type="System.String" Value="OBX" />

<Element Name="OBRLabel" Type="System.String" Value="LabResult\_OBX\_ExternalId" />

<Element Name="SpringDay" Type="System.String" Value="01/01/2011" />

<Element Name="Domain" Type="System.String" Value="ABC" />

</DataConstants>

<!--Section IV, user defined the data transform for each elements in Section II -->

<!-- Section VI, Plugin Configuration -->

<Plugins>

<!-- Here, we can define multiple transform library, so that user can import their library. We can treat each library as a plugin-->

<Plugin Name="MessageHelper" Assembly="Microsoft.Health.Pipeline.Transforms, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396" ClassName ="MessageHelpers">

<NameSpace>Microsoft.Health.Pipeline.Transforms</NameSpace>

<URL>./Microsoft.Health.Pipeline.Transforms.dll</URL>

<SourceType>FileSystem</SourceType>

<Methods>

<Method Name="Concatenate" AssemblyMethodName="Concatenate">

<Parameters>

<MethodParameter Name="Separator" Type="System.String"/>

<MethodParameter Name="FirstString" Type="System.String"/>

<MethodParameter Name="LastString" Type="System.String"/>

</Parameters>

</Method>

<Method Name="ToTitleCase" AssemblyMethodName="ToTitleCase">

<Parameters>

<MethodParameter Name="LowCaseString" Type="System.String"/>

</Parameters>

</Method>

<Method Name="ConcatenateChildren" AssemblyMethodName="ConcatenateChildren">

<Parameters>

<MethodParameter Name="fTable" Type="Microsoft.Health.Pipeline.Model.FirstGenerationTable" Source="Microsoft.Health.Pipeline.Model, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396"/>

<MethodParameter Name="dTable" Type="Microsoft.Health.Pipeline.Model.DataParserTable" Source="Microsoft.Health.Pipeline.Model, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396"/>

<MethodParameter Name="childTableName" Type="System.String"/>

<MethodParameter Name="label" Type="System.String"/>

</Parameters>

</Method>

</Methods>

</Plugin>

</Plugins>

<EnterpriseId>

<EnterpriseIdAction Action="StandardMapping" AnchorTable="PID">

<InputIdentifiers>

<SourceIdentifier Name="patient">

<Id Name="Id" Type="System.String" SourceType="DerivedField" SourceTable="PID" SourceName="FullName" />

<Domain Name="Domain" Type="System.String" SourceType="Constant" SourceName="Domain" />

</SourceIdentifier>

<SourceIdentifier Name="encounter">

<Id Name="Id" Type="System.String" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_5\_1" />

<Domain Name="Domain" Type="System.String" SourceType="Constant" SourceName="Domain" />

</SourceIdentifier>

</InputIdentifiers>

</EnterpriseIdAction>

</EnterpriseId>

</DataParserConfiguration>

]]>

</DefaultValue>

</Setting>

<Setting Name="PluginConfigurationXml" Type="System.String">

<DefaultValue>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

## AmalgaID

AmalgaID walkthrough example

High level steps

1. Define a G1 message shredder to shred ADT.G data from Model System Queue
   1. Define a DerivedElement for MRN from the PID table
   2. Define context keys for Patient\_PID\_ExternalId and Account\_PID\_ExternalId

//G1 Service XML

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

xmlns:i ="http://www.microsoft.com/2011/03/health/deployment/internal"

Type = "WindowsWorker"

Name="FirstGenerationParser"

InstanceName="MessageShredder\_ADT\_G"

Id="2684661F-AAE0-40C1-9AC9-2D8061CEA8C6"

i:Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName="FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="FirstGenerationParser.exe" />

<PerformanceCategories>

<PerformanceCategory Name="Amalga Message Shredder Service Stats" Version="" CodeName="MessageShredder" Help="Store the stats for the Message Shredder" MultiInstance="Yes">

<PerformanceCounter Name="Messages Received" CodeName="MessagesReceived" Type="NumberOfItems64" Help="Records the the total number of messages received by the Message Shredder" />

<PerformanceCounter Name="Messages Per Second" CodeName="MessagesProcessedPerSec" Type="RateOfCountsPerSecond32" Help="Records the number of messages received per second" />

<PerformanceCounter Name="Messages Successful" CodeName="MessagesSuccessful" Type="NumberOfItems64" Help="Records the total number of messages processed successfully" />

<PerformanceCounter Name="Messages Unsuccessful" CodeName="MessagesUnsuccessful" Type="NumberOfItems64" Help="Records the number of messages that failed during processing" />

</PerformanceCategory>

</PerformanceCategories>

<Settings>

<Setting Name="L1DatabaseConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=ModelSystemQueue;uid=sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="G1DatabaseConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!;Connection Timeout=60</DefaultValue>

</Setting>

<Setting Name="G1DatabaseName" Type="System.String">

<DefaultValue>ClinicalData</DefaultValue>

</Setting>

<Setting Name="DatabaseSchema" Type="System.String">

<DefaultValue>ADT\_G</DefaultValue>

</Setting>

<Setting Name="FeedName" Type="System.String">

<DefaultValue>ADT.G</DefaultValue>

</Setting>

<Setting Name="LastProcessedSequenceNumber" Type="System.Int32">

<DefaultValue>0</DefaultValue>

</Setting>

<Setting Name="BatchSize" Type="System.Int32">

<DefaultValue>250</DefaultValue>

</Setting>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[

<?xml version="1.0" encoding="utf-8"?>

<FirstGenerationConfiguration>

<Segments>

<Segment Name="PID" SourceName="PID" Description="Person Identification">

<DerivedElements>

<DerivedElement Name="MRN" Description="Patient MRN" SourceName="PID\_3\_Repeating" Filter="/Repetitions/Repetition[2]/PID-3-1">

<ConceptId>565F4011-15F7-4D48-823F-BB6775B06B7B</ConceptId>

</DerivedElement>

</DerivedElements>

<ContextKeys>

<ContextKey>

<Name>Patient\_PID\_ExternalId</Name>

<ConceptId>F76DACCC-08F4-4266-B880-28D55487FD99</ConceptId>

<IsPrimary>true</IsPrimary>

<SourceKeys>

<SourceKey SourceTable="PID" SourceName="MRN" Description="MRN from the second repetition." />

</SourceKeys>

</ContextKey>

<ContextKey>

<Name>Account\_PID\_ExternalId</Name>

<ConceptId>F9CE1E7F-925E-40BF-BE84-BF2CFBD08B71</ConceptId>

<IsPrimary>false</IsPrimary>

<SourceKeys>

<SourceKey SourceTable="PID" SourceName="PID\_18\_1" Description="Account" />

</SourceKeys>

</ContextKey>

</ContextKeys>

</Segment>

<Segment Name="PV1" SourceName="PV1" ParentName="PID" Description="Patient Visit">

<DerivedElements />

<ContextKeys />

</Segment>

</Segments>

</FirstGenerationConfiguration>

]]>

</DefaultValue>

</Setting>

<Setting Name="ShredderPluginConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[<Plugin Name="MessageHelper" Assembly="Microsoft.Health.Pipeline.FirstGenerationShredders, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396" ClassName ="HL7Shredder">

<NameSpace>Microsoft.Health.Pipeline.FirstGenerationShredders</NameSpace>

<URL>C:\Program Files\Microsoft Amalga\System 3.0\Application Worker Services\MessageShredder\_ADT\_G\Microsoft.Health.Pipeline.FirstGenerationShredders.dll</URL>

<SourceType>FileSystem</SourceType>

</Plugin>]]>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

1. Define a G2 data parser to get First Name, Last Name from the G1 PID table, and concatenate them into a Full Name field in a G2 derived table.
2. Define AmalgaID section in G2 config

//G2 Service XML (complete)

<?xml version="1.0" encoding="utf-8" ?>

<DeploymentManifest xmlns="http://www.microsoft.com/2011/03/health/deployment/hosting"

Type = "WindowsWorker"

Name="DataParserApplication\_Ins1"

InstanceName="DataParserService\_ADT\_G"

Id="44162965-1f39-4f81-94b4-63002da90b90"

Version ="1.0.0.0"

>

<WindowsWorkerManifest LogicalHostName="FabricLogicalHost" AuthenticationMode ="None" ClusteringMode="NLB" HostAssembly="Microsoft.Health.Pipeline.SecondGeneration.exe"/>

<PerformanceCategories>

<PerformanceCategory Name="Amalga Data Parser Service Stats" Version="" CodeName="DataParser" Help="Store the stats for the Data Parser" MultiInstance="Yes">

<PerformanceCounter Name="Messages Received" CodeName="MessagesReceived" Type="NumberOfItems64" Help="Records the the total number of messages received by the Data Parser" />

<PerformanceCounter Name="Messages Per Second" CodeName="MessagesReceivedSec" Type="RateOfCountsPerSecond32" Help="Records the number of messages received per second" />

<PerformanceCounter Name="Messages Successful" CodeName="MessagesProcessedSuccessful" Type="NumberOfItems64" Help="Records the total number of messages processed successfully" />

<PerformanceCounter Name="Messages Unsuccessful" CodeName="MessagesProcessedUnSuccessful" Type="NumberOfItems64" Help="Records the number of messages that failed during processing" />

</PerformanceCategory>

<PerformanceCategory Name="Amalga Enterprise ID" Version="" CodeName="AmalgaIdPerfCounters" Help="Amalga Enterprise Id Processor Counters" MultiInstance="Yes">

<PerformanceCounter Name="AmalgaEnterpriseId Messages Total" CodeName="MessagesTotal" Type="numberOfItems64" Help="The total # of messages run through AmalgaID"/>

<PerformanceCounter Name="AmalgaEnterpriseId Lookup Total" CodeName="LookupTotal" Type="numberOfItems64" Help="The total # of matching lookups performed"/>

<PerformanceCounter Name="AmalgaEnterpriseId Deadlock Total" CodeName="DeadlockTotal" Type="numberOfItems64" Help="The total # of submissions that were SQL deadlocked"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches EID" CodeName="MatchesEID" Type="numberOfItems64" Help="The # of lookups that result in an EID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches GEID" CodeName="MatchesGEID" Type="numberOfItems64" Help="The # of lookups that result in an GEID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches SOID" CodeName="MatchesSOID" Type="numberOfItems64" Help="The # of lookups that result in an SOID match"/>

<PerformanceCounter Name="AmalgaEnterpriseId Matches OID" CodeName="MatchesOID" Type="numberOfItems64" Help="The # of lookups that result in an OID match"/>

</PerformanceCategory>

</PerformanceCategories>

<Settings>

<Setting Name="ClinicalDatabaseConnectionString" Type="System.String">

<!--TODO: change this back to az-sa, and grant appropriate authorization-->

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="ClinicalDatabaseAdminConnectionString" Type="System.String">

<DefaultValue>server=localhost;database=ClinicalData;uid=sa;pwd=passw0rd!</DefaultValue>

</Setting>

<Setting Name="DatabaseSchema" Type="System.String">

<DefaultValue>ADT\_G</DefaultValue>

</Setting>

<Setting Name="FeedName" Type="System.String">

<DefaultValue>ADT.G</DefaultValue>

</Setting>

<Setting Name="LastProcessedSequenceNumber" Type="System.Int32">

<DefaultValue>0</DefaultValue>

</Setting>

<Setting Name="BatchSize" Type="System.Int32">

<DefaultValue>250</DefaultValue>

</Setting>

<Setting Name="FeedConfigurationXml" Type="System.String">

<DefaultValue>

<![CDATA[

<?xml version="1.0" encoding="utf-8" ?>

<DataParserConfiguration>

<!--Section I, user defines the input message structure. -->

<MessageShredderInput>

<Segment Name="PID" SourceName="PID">

<Elements>

<Element Name="PID\_5\_1" />

<Element Name="PID\_5\_2" />

<Element Name="PID\_3\_1" />

<Element Name="PID\_3\_5" />

<Element Name="PID\_18\_1" />

<Element Name="MRN" />

</Elements>

</Segment>

</MessageShredderInput>

<!-- Section II, user defines the output message structure-->

<DataParserOutput>

<Segment Name="PID" SourceName="PID">

<Elements>

<Element Name="FName" Type="System.String" MaxLength="50">

<ConceptId>11627229-A030-41C4-B323-9C5B8DAE0384</ConceptId>

<Transform>

<Action>ToTitleCase</Action>

<InputParameters>

<Parameter Name="LowCaseString" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_5\_1" />

</InputParameters>

</Transform>

</Element>

<Element Name="LName" Type="System.String" MaxLength="50">

<ConceptId>762F8617-40E0-4590-85F0-589AA98D6016</ConceptId>

<Transform>

<Action>ToTitleCase</Action>

<InputParameters>

<Parameter Name="LowCaseString" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_5\_2" />

</InputParameters>

</Transform>

</Element>

<Element Name="FullName" Type="System.String" MaxLength="150">

<ConceptId>4CF03880-FCA7-49DE-A396-345B480E7441</ConceptId>

<Transform>

<Action>Concatenate</Action>

<InputParameters>

<Parameter Name="Separator" SourceType="Constant" SourceName="Space" />

<Parameter Name="FirstName" SourceType="DerivedField" SourceTable="PID" SourceName="FName" />

<Parameter Name="LastName" SourceType="DerivedField" SourceTable="PID" SourceName="LName" />

</InputParameters>

</Transform>

</Element>

</Elements>

</Segment>

</DataParserOutput>

<ExtensibleDataParserOutput>

<Segment Name="PID" SourceName="PID">

<ConceptId>16452154-0BE5-47BD-A527-28A244D3EDD0</ConceptId>

<Elements>

<Element Name="Gender" Type="System.String" MaxLength="50" />

</Elements>

</Segment>

</ExtensibleDataParserOutput>

<!--Section III, user defines the constants, which will be used in G2 parsing.-->

<DataConstants>

<Element Name="Space" Type="System.String" Value=" " />

<Element Name="IndependenceDay" Type="System.DateTime" Value="07/07/2011" />

<Element Name="OBRChildTable" Type="System.String" Value="OBX" />

<Element Name="OBRLabel" Type="System.String" Value="LabResult\_OBX\_ExternalId" />

<Element Name="SpringDay" Type="System.String" Value="01/01/2011" />

<Element Name="Domain" Type="System.String" Value="ABC" />

</DataConstants>

<!--Section IV, user defined the data transform for each elements in Section II -->

<!-- Section VI, Plugin Configuration -->

<Plugins>

<!-- Here, we can define multiple transform library, so that user can import their library. We can treat each library as a plugin-->

<Plugin Name="MessageHelper" Assembly="Microsoft.Health.Pipeline.Transforms, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396" ClassName ="MessageHelpers">

<NameSpace>Microsoft.Health.Pipeline.Transforms</NameSpace>

<URL>./Microsoft.Health.Pipeline.Transforms.dll</URL>

<SourceType>FileSystem</SourceType>

<Methods>

<Method Name="Concatenate" AssemblyMethodName="Concatenate">

<Parameters>

<MethodParameter Name="Separator" Type="System.String"/>

<MethodParameter Name="FirstString" Type="System.String"/>

<MethodParameter Name="LastString" Type="System.String"/>

</Parameters>

</Method>

<Method Name="ToTitleCase" AssemblyMethodName="ToTitleCase">

<Parameters>

<MethodParameter Name="LowCaseString" Type="System.String"/>

</Parameters>

</Method>

<Method Name="ConcatenateChildren" AssemblyMethodName="ConcatenateChildren">

<Parameters>

<MethodParameter Name="fTable" Type="Microsoft.Health.Pipeline.Model.FirstGenerationTable" Source="Microsoft.Health.Pipeline.Model, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396"/>

<MethodParameter Name="dTable" Type="Microsoft.Health.Pipeline.Model.DataParserTable" Source="Microsoft.Health.Pipeline.Model, Version=3.0.0.0, Culture=neutral, PublicKeyToken=a03c60abe3a7c396"/>

<MethodParameter Name="childTableName" Type="System.String"/>

<MethodParameter Name="label" Type="System.String"/>

</Parameters>

</Method>

</Methods>

</Plugin>

</Plugins>

<EnterpriseId>

<EnterpriseIdAction Action="StandardMapping" AnchorTable="PID">

<InputIdentifiers>

<SourceIdentifier Name="Patient">

<Id Name="Id" Type="System.String" SourceType="ShreddedInput" SourceTable="PID" SourceName="MRN" />

<Domain Name="Domain" Type="System.String" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_3\_5" />

</SourceIdentifier>

<SourceIdentifier Name="Encounter">

<Id Name="Id" Type="System.String" SourceType="ShreddedInput" SourceTable="PID" SourceName="PID\_18\_1" />

<Domain Name="Domain" Type="System.String" SourceType="Constant" SourceName="Domain" />

</SourceIdentifier>

</InputIdentifiers>

</EnterpriseIdAction>

</EnterpriseId>

</DataParserConfiguration>

]]>

</DefaultValue>

</Setting>

<Setting Name="PluginConfigurationXml" Type="System.String">

<DefaultValue>

</DefaultValue>

</Setting>

</Settings>

</DeploymentManifest>

1. Do manual binding of the Patient Entity by creating two defining views
   1. v\_Patient
   2. v\_Patient\_Collapsed

//SQL for v\_Patient

USE [ClinicalData]

GO

/\*\*\*\*\*\* Object: View [dbo].[v\_Patient] Script Date: 08/18/2011 10:27:29 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[v\_Patient]

AS

SELECT aEID.OID as Person\_ID, AEID.SOID as Patient\_ID,

PID\_Derived.FName as PatientGIvenName,

PID\_Derived.LName as PatientFamilyName,

PID.PID\_5\_3 as PatientMiddleName,

NULL as PatientPrefixName,

NULL as PatientSuffixName,

PID.PID\_7 as DateTimeOfBirth,

PID.PID\_8 as Gender,

PID.PID\_11\_1 as PatientStreetAddress,

PID.PID\_11\_2 as PatientAddressOtherDesignations,

PID.PID\_11\_3 as PatientAddressCity,

PID.PID\_11\_4 as PatientAddressStateOrProvince,

PID.PID\_11\_5 as PatientAddressPostalCode,

PID.PID\_11\_6 as PatientAddressCountry,

'Encounter' as PatientLevel,

NULL as ContactInfo\_Home,

NULL as contactInfo\_Business,

PID.PID\_10 as PatientRace,

PID.SequenceNumber as SequenceNumber

FROM ADT\_G.PID PID

LEFT JOIN ADT\_G.PID\_Derived PID\_Derived ON PID.RowId = PID\_Derived.RowId

LEFT JOIN AmalgaId.AEID202 AEID ON AEID.GEIDexternal = PID.PID\_18\_1 -- AND AEID.IDdomain = 'ABC'

GO

//SQL for v\_Patient\_Collapsed

USE [ClinicalData]

GO

/\*\*\*\*\*\* Object: View [dbo].[v\_Patient\_Collapsed] Script Date: 08/18/2011 10:28:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[v\_Patient\_Collapsed]

AS

WITH CTE AS

(

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY Patient\_ID ORDER BY sequenceNumber DESC) AS RowN

FROM V\_Patient

)

SELECT \* FROM CTE WHERE RowN=1

GO