AiL Developer Guide

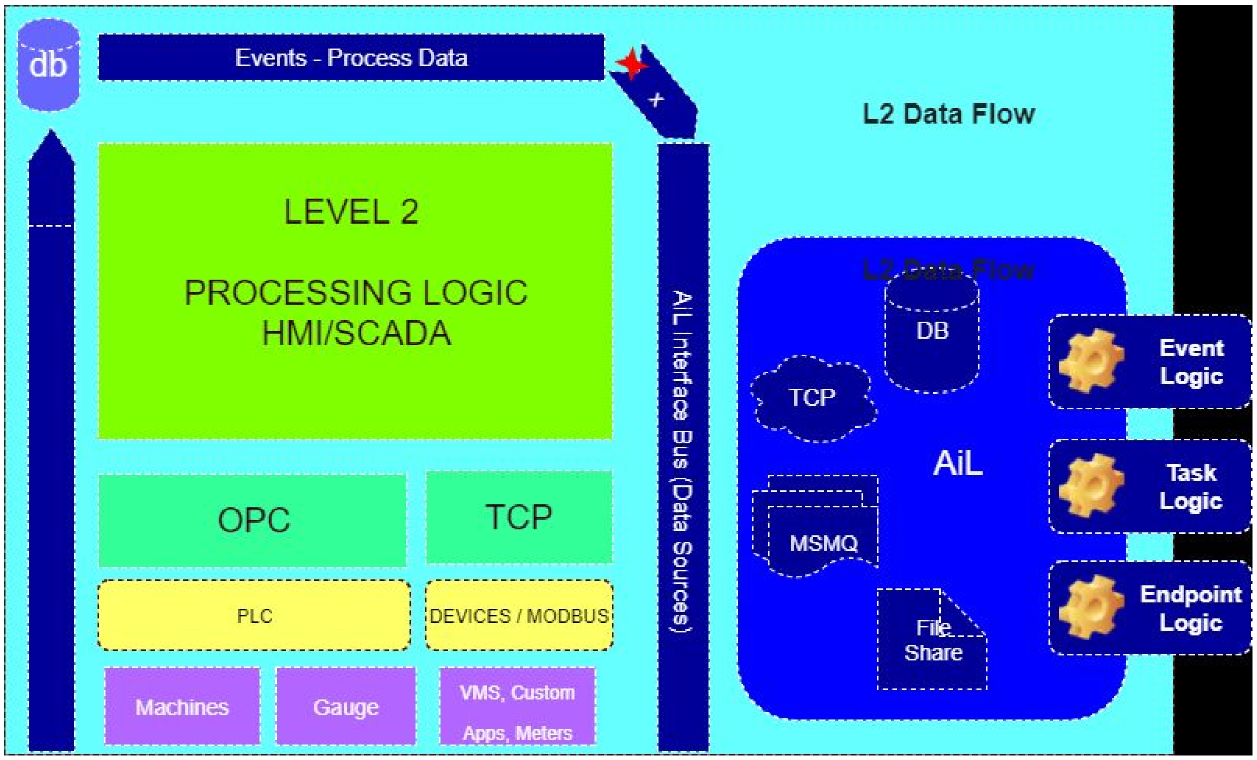
Revision 3.0

Introduction

What is AiL?

Automation Interface Layer (AiL) is an enterprise tool that allows building a serverless computing environment by allowing the developer to create integration logic in script form.

The core strength of AiL is demonstrated when integrating messaging services between two decoupled systems. Two software components that make up AIL is an editor IDE (Integrated Development Environment) and windows service (Script Execution Service). The service is used to execute scripts that can be used as integration points between two different systems. AiL can execute ETL (Extract Translate and Load) code, utility functions (timed) or application level functions with I/O.



The service can run in desktop mode or as a windows service under the service control manager (SCM). Running the service in desktop mode is not recommended in production deployments but is useful for development and testing scenarios.

Event Message Processing

When and how events are processed can be determined based on configuration or invocation. The below charts describe the script types, available methods and return statuses available in the AiL.

Script Type Definitions

Type	Description	Invocation Methods
Event	Event scripts are executed on-demand based on input from an external system	From a Registered Data Source Event/Task Script Direct insert of a record into the message table of the AiL database.
Task	Task scripts are executed on a timed interval or a daily schedule	Internal Timer
Endpoint	Endpoint scripts are fired via a REST interface. Endpoint scripts can accept arguments and are also capable of returning a JSON result back to the calling program. Endpoint scripts allow the AiL service to execute a script in a synchronous fashion.	RESTful Call

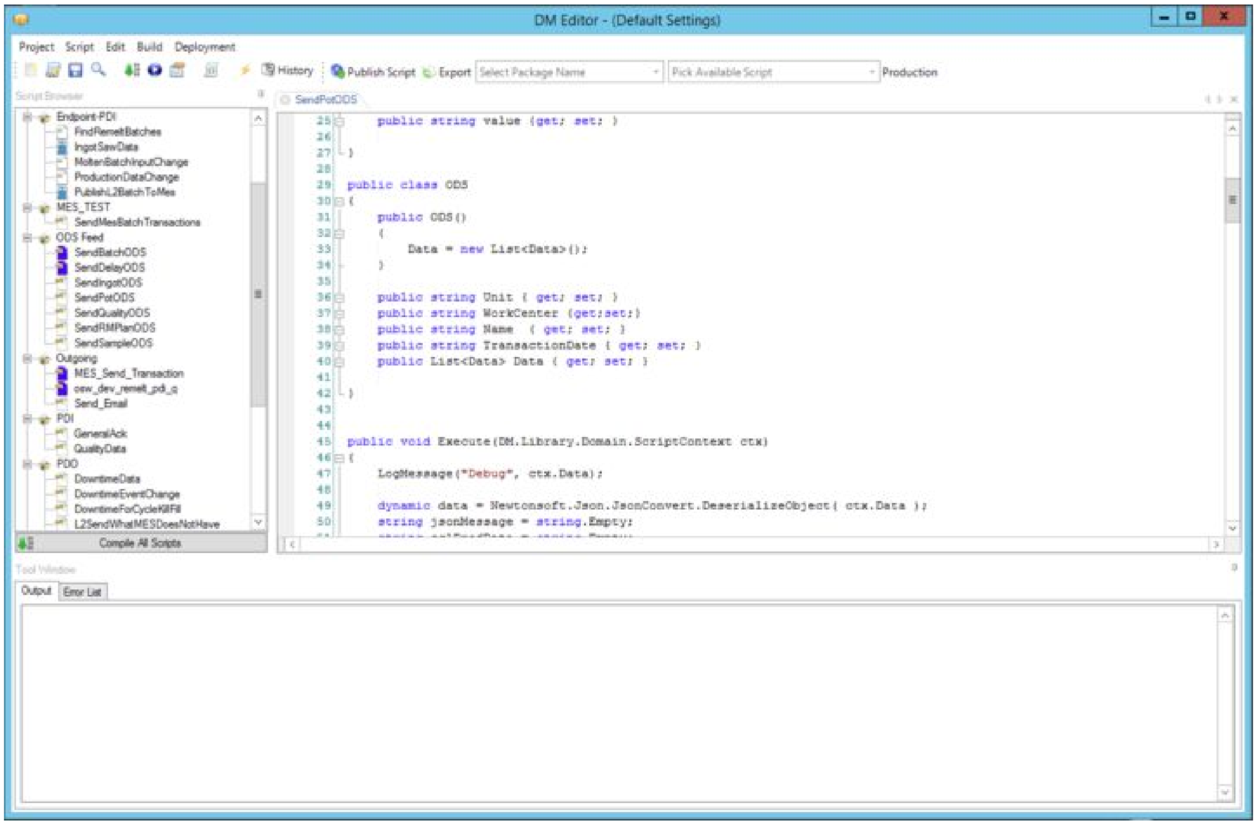
Execution Context Methods

Name	Description
Thread Pool (TP)	Messages placed in the Thread Pool can be processed at any time and in any order. If an event message is considered to be intrinsic of any other message, the best place to put this message would be in the Thread Pool. Messages returning a status of ‘Blocking’ in the Thread Pool is not valid.
Primary Sync Queue (PSQ)	Messages can be placed in the PSQ by checking the “Synchronous Message” flag found within the script properties. Messages in the PSQ will be processed in the order which it was placed the queue. Messages returning a status of ‘Blocking’ in the PSQ is not valid.
Named Sync Queue (NSQ)	Messages can be placed in an NSQ by entering a name into the “TargetSyncQueue” found within the script properties. Any valid alphanumeric value can be used for the “TargetSyncQueue”. A return Status of Blocking is a valid status for an NSQ. A return status of Failed or FailedAck will cause an NSQ to be placed into a locked state. Newly received message and remaining messages within the queue will be placed in Cancelled status. Intervention is required by an AiL Administrator to release the locked NSQ and reprocess messages as needed.
Dynamic Sync Queue (DSQ)	Messages can be placed in a DSQ by passing the “TargetQueueName” as an argument when creating the event message. Any valid alphanumeric value can be used for the “TargetQueueName”. A return Status of Blocking is a valid status for an DSQ. A return status of Failed or FailedAck will cause an DSQ to be placed into a locked state. Newly received message and remaining messages within the queue will be placed in Cancelled status. Intervention is required by an AiL Administrator to release the locked DSQ and reprocess messages as needed.

Event Return Status

Name	Description
NotProcessed	Script has been received but not loaded in the AiL service.
Processing	Script is in the queuing system of AiL ready to be executed.
Executing	Script is actively executing script logic.
PendingAck	Script has executed with no errors but is waiting for an external system to acknowledge success.
Complete	Script has completed all processing with success
CompleteAck	Script has received a transaction acknowledgement and is complete with success.
Failed	Script experienced a failure.
FailedAck	Script executed with no errors but the acknowledgment from an external system experienced a failure.
Blocking	Script could not complete execution but requests to be re-executed in the future. Blocking scripts are run outside of the thread pool in a queued fashion. Blocking scripts will prevent any predecessors for executing until it Cancels or Completes.
Cancelled	Script attempted to execute but could not and was forced into cancellation by the script author.
Aborted	Detected script executions that could not be completed and will require manual intervention to re-execute or delete.
Delete	Will be marked for removal.

AiL Editor (Integrated Development Environment)

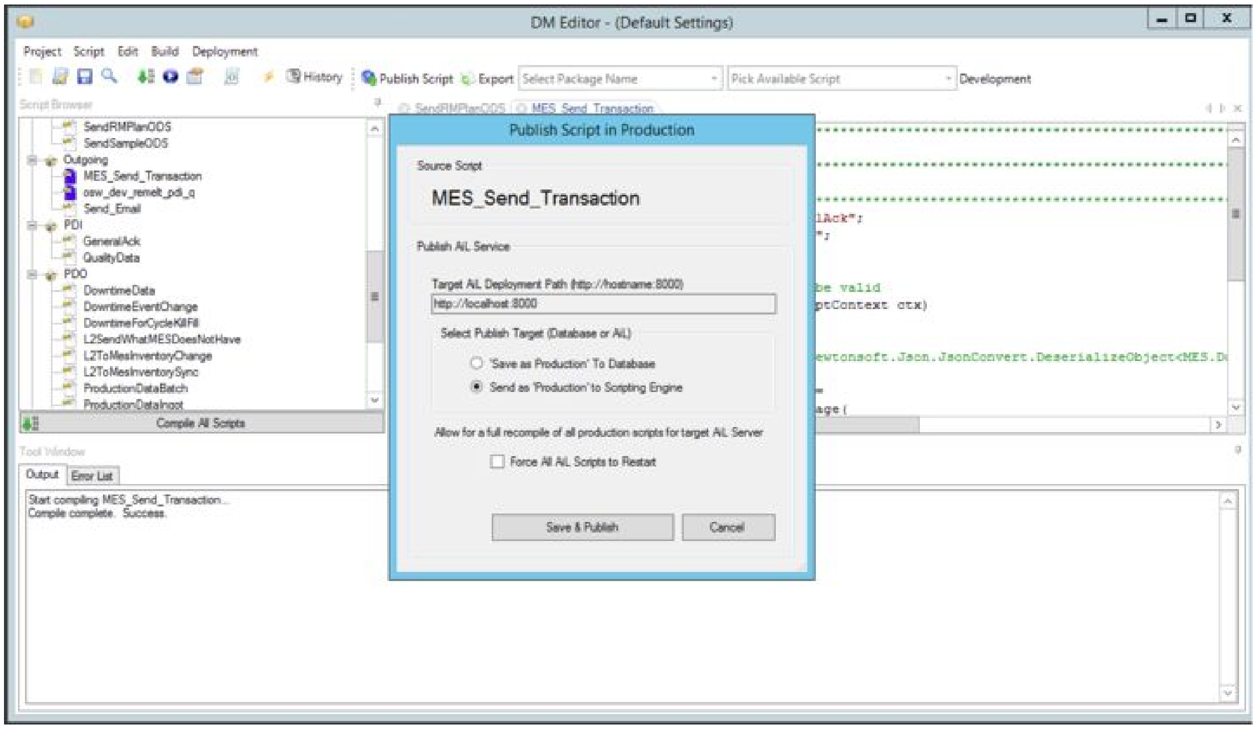


```
    public string value {get; set; }

}

public class ODS
{
    public ODS()
    {
        Data = new List<Data>();
    }

    public string Unit { get; set; }
    public string WorkCenter {get;set;}
    public string Name  { get; set; }
    public string TransactionDate { get; set; }
    public List<Data> Data { get; set; }

}


public void Execute(DM.Library.Domain.ScriptContext ctx)
{
    LogMessage("Debug", ctx.Data);

    dynamic data = Newtonsoft.Json.JsonConvert.DeserializeObject( ctx.Data );
    string jsonMessage = string.Empty;
```

The development environment provides a way to create scripts (event, task and endpoint) within the AIL environment.

Use packages provides a way to organize the scripts for ease of use. Scripts can take on two different states: Development or Production.   When a script is in the production state it is considered to be the active script within the AiL service.   When a script is considered development it may be an older version or most likely the most recently edited version. When a script is currently in a production script and saved within the IDE, a new version is created in the development state.

Script Publish



Source Script: MES_Send_Transaction

Publish AiL Service

Target AiL Deployment Path (http://hostname:8000): http://localhost:8000

Select Publish Target (Database or AiL)

- ○ 'Save as Production' To Database
- ● Send as 'Production' to Scripting Engine

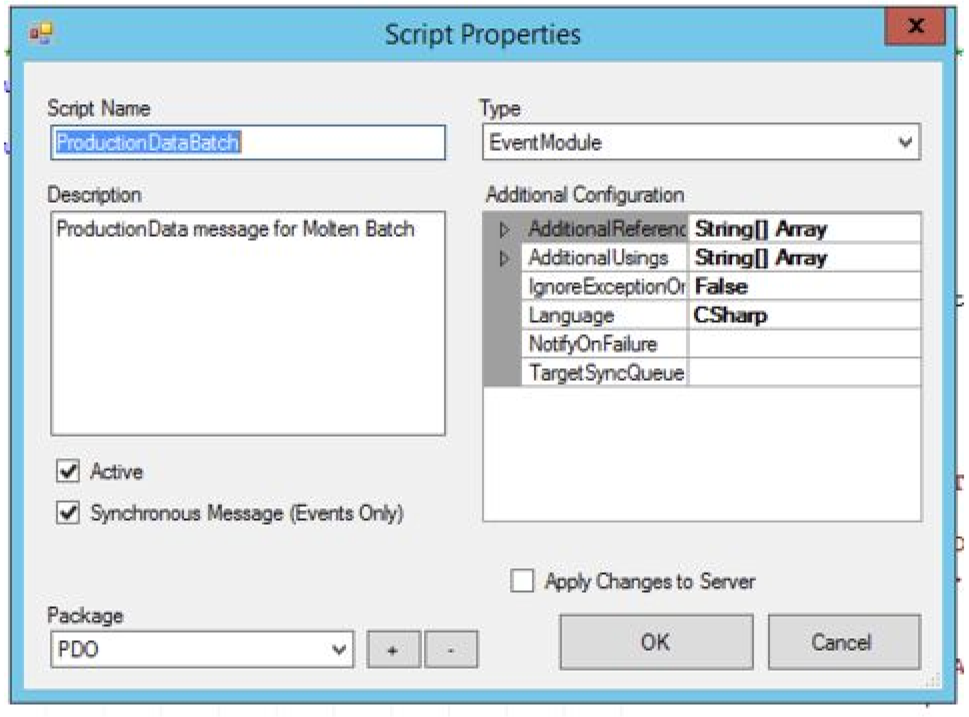
Allow for a full recompile of all production scripts for target AiL Server

- ☐ Force All AiL Scripts to Restart

```
Start compiling MES_Send_Transaction...
Compile complete.  Success.
```

To move scripts from a the state of development to production, requires a publish action. Two ways to publish a script can be to the database or the actual scripting engine. When making a script production in the database only, it will not update the active script until the service is restarted. Once the service restarts then the script will be picked up and made active. If publishing to the scripting engine then the new script changes will be executed immediately. Any non-executing scripts will be used with the new version.

Script Definitions


Script Properties

Script Name: ProductionDataBatch

Type: EventModule

Description: ProductionData message for Molten Batch

Additional Configuration

AdditionalReferenc	String[] Array
AdditionalUsings	String[] Array
IgnoreExceptionOr	False
Language	CSharp
NotifyOnFailure	
TargetSyncQueue	

[✓] Active

[✓] Synchronous Message (Events Only)

[ ] Apply Changes to Server

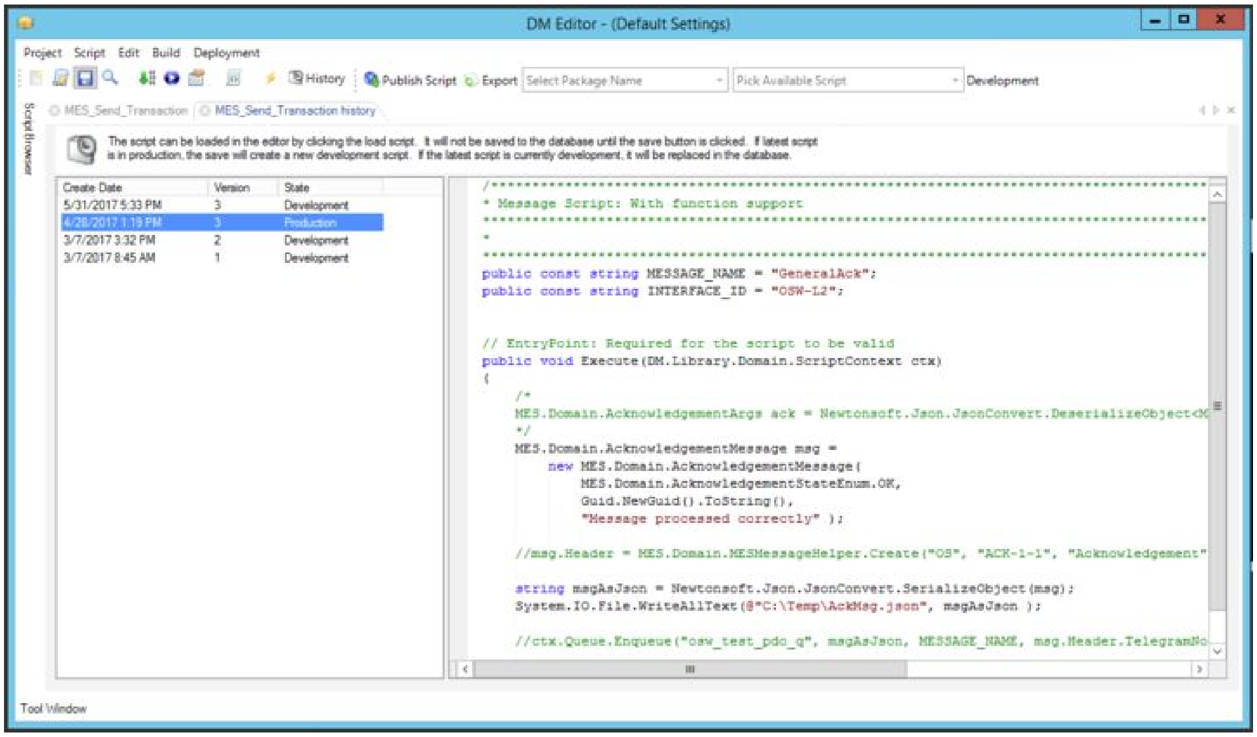
Package: PDO

Script definitions are assigned from the script properties. Use this screen to define how the script will be used within the system.

Property	Description
Name	Assigns the name of the script to the execution context
Type	Defines if the script will be Event, Endpoint or Task
Active	Determines if the script should be executed or ignored by the system. Scripts unchecked as Active will not be executed by the service in any fashion.
Synchronous Message (Events Only)	Indicates that the message will be placed on a Primary Sync Queue (PSQ) for ordered processing. If the field TargetSyncQueue is set to a name, then the script will be executed on a Named Sync Queue (NSQ).
AdditionalReferences	Allows external .NET assemblies to be included for access (third party or custom assembly code be placed here).
AdditionalUsings	Include namespaces within your script definition for ease of access to built in class names within the .NET framework or your custom code.

IgnoreExceptionOnSyncQueue	When set to false, if an exception or error is detected, the NQ or DQ will be suspended from processing any additional messages. When set to true, the exception or error is noted but messages following will continue to be processed.
Language	Defines the .NET language used for the script. Can be CSharp or VBNet.
NotifyOnFailure	Comma delimited list of email to notify whenever a failure message happens at any time.
TargetQueue	Names a target queue to be processed at the Script Definition only. Requires the Sync flag to be checked to be active.

Script History



The script can be loaded in the editor by clicking the load script. It will not be saved to the database until the save button is clicked. If latest script is in production, the save will create a new development script. If the latest script is currently development, it will be replaced in the database.

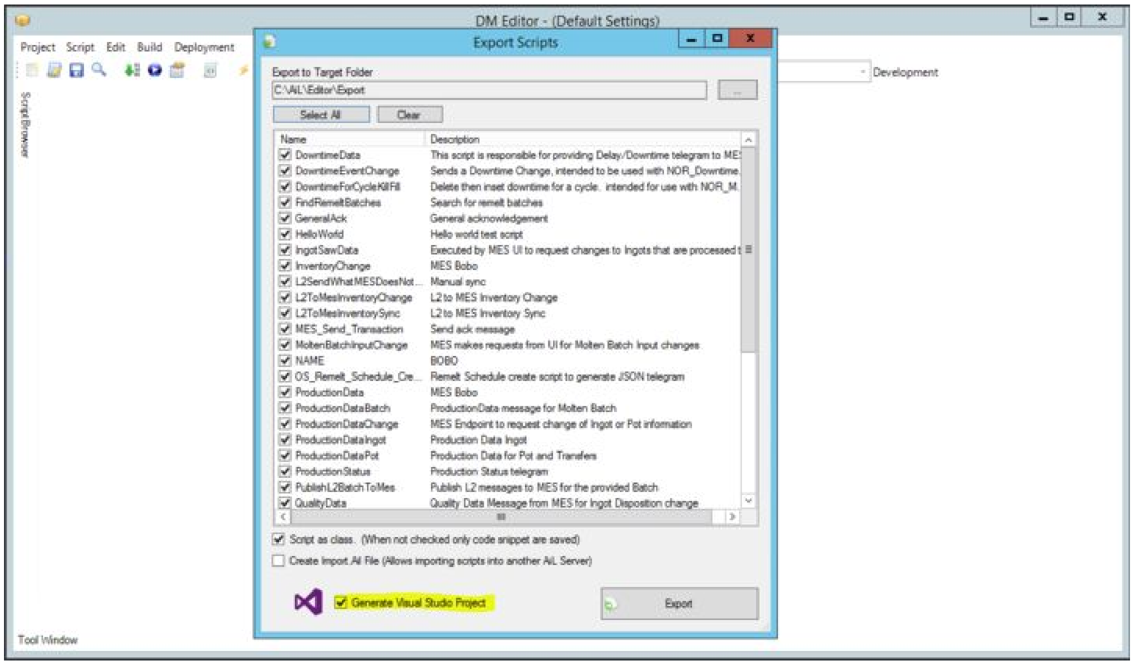
Create Date	Version	State
5/31/2017 5:33 PM	3	Development
4/28/2017 1:19 PM	3	Production
3/7/2017 3:32 PM	2	Development
3/7/2017 8:45 AM	1	Development

```
/*****************************************************************************
* Message Script: With function support
*****************************************************************************
*
*****************************************************************************
public const string MESSAGE_NAME = "GeneralAck";
public const string INTERFACE_ID = "OSW-L2";


// EntryPoint: Required for the script to be valid
public void Execute(DM.Library.Domain.ScriptContext ctx)
{
    /*
    MES.Domain.AcknowledgementArgs ack = Newtonsoft.Json.JsonConvert.DeserializeObject<M
    */
    MES.Domain.AcknowledgementMessage msg =
        new MES.Domain.AcknowledgementMessage(
            MES.Domain.AcknowledgementStateEnum.OK,
            Guid.NewGuid().ToString(),
            "Message processed correctly" );

    //msg.Header = MES.Domain.MESMessageHelper.Create("OS", "ACK-1-1", "Acknowledgement"

    string msgAsJson = Newtonsoft.Json.JsonConvert.SerializeObject(msg);
    System.IO.File.WriteAllText(@"C:\Temp\AckMsg.json", msgAsJson );

    //ctx.Queue.Enqueue("osw_test_pdo_q", msgAsJson, MESSAGE_NAME, msg.Header.TelegramNo
```

History of script changes can be tracked by using the history viewer. The history viewer provides a way to see what scripts changes have taken place.

Old versions of a script can be made active within the environment. When making an old version of the script active, the AiL editor will create a new version of the script from the previous version selected. The script history is considered immutable within the system and old scripts will never be removed.

Script Export/Import



Export to Target Folder
C:\AiL\Editor\Export

Name	Description
✓ DowntimeData	This script is responsible for providing Delay/Downtime telegram to ME
✓ DowntimeEventChange	Sends a Downtime Change, intended to be used with NOR_Downtime
✓ DowntimeForCycleKillFill	Delete then inset downtime for a cycle. intended for use with NOR_M
✓ FindRemeltBatches	Search for remelt batches
✓ GeneralAck	General acknowledgement
✓ HelloWorld	Hello world test script
✓ IngotSawData	Executed by MES UI to request changes to Ingots that are processed t
✓ InventoryChange	MES Bobo
✓ L2SendWhatMESDoesNot...	Manual sync
✓ L2ToMesInventoryChange	L2 to MES Inventory Change
✓ L2ToMesInventorySync	L2 to MES Inventory Sync
✓ MES_Send_Transaction	Send ack message
✓ MoltenBatchInputChange	MES makes requests from UI for Molten Batch Input changes
✓ NAME	BOBO
✓ OS_Remelt_Schedule_Cre...	Remelt Schedule create script to generate JSON telegram
✓ ProductionData	MES Bobo
✓ ProductionDataBatch	ProductionData message for Molten Batch
✓ ProductionDataChange	MES Endpoint to request change of Ingot or Pot information
✓ ProductionDataIngot	Production Data Ingot
✓ ProductionDataPot	Production Data for Pot and Transfers
✓ ProductionStatus	Production Status telegram
✓ PublishL2BatchToMes	Publish L2 messages to MES for the provided Batch
✓ QualityData	Quality Data Message from MES for Ingot Disposition change

✓ Script as class. (When not checked only code snippet are saved)

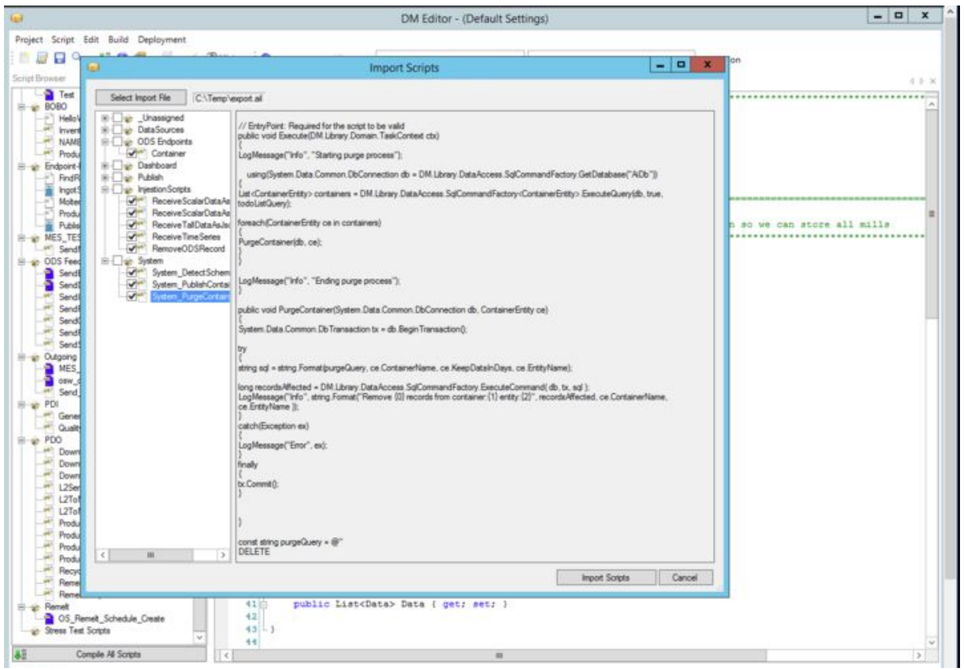
☐ Create Import All File (Allows importing scripts into another AiL Server)

✓ Generate Visual Studio Project

Use the export tool to make a backup of the scripts currently being developed. Use this backup method as a way to place scripts into another version control system or to copy to another instance of AiL.

Export also provides a way to create an integration with Microsoft Visual Studio by generating a test project used for debugging and troubleshooting of scripts.

Example of the Import Dialog



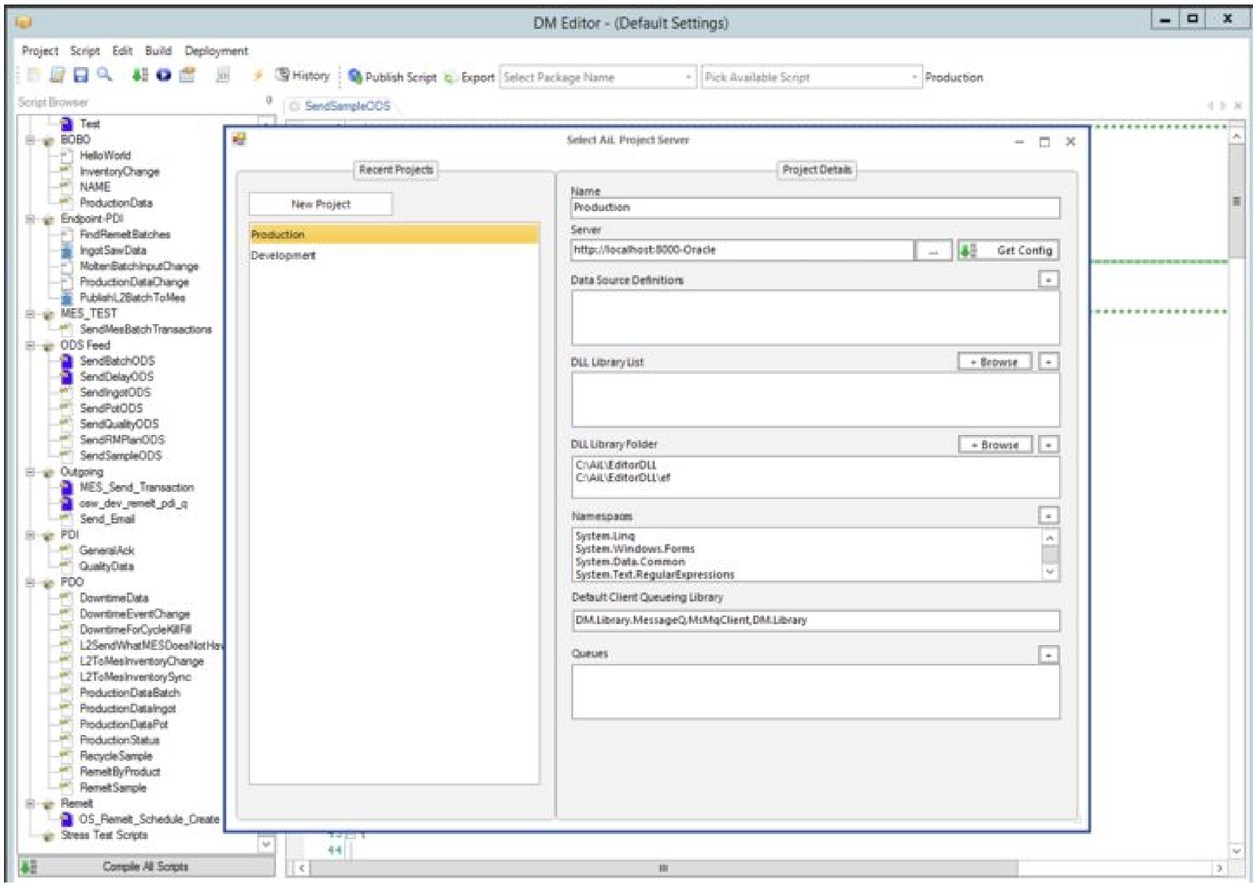
C:\Temp\export.ail

```
// EntryPoint: Required for the script to be valid
public void Execute(DM.Library.Domain.TaskContext ctx)
{
LogMessage("Info", "Starting purge process");

   using(System.Data.Common.DbConnection db = DM.Library.DataAccess.SqlCommandFactory.GetDatabase("AiDb"))
{
List<ContainerEntity> containers = DM.Library.DataAccess.SqlCommandFactory<ContainerEntity>.ExecuteQuery(db, true,
todoListQuery);

foreach(ContainerEntity ce in containers)
{
PurgeContainer(db, ce);
}
}

LogMessage("Info", "Ending purge process");
}

public void PurgeContainer(System.Data.Common.DbConnection db, ContainerEntity ce)
{
System.Data.Common.DbTransaction tx = db.BeginTransaction();

try
{
string sql = string.Format(purgeQuery, ce.ContainerName, ce.KeepDataInDays, ce.EntityName);

long recordsAffected = DM.Library.DataAccess.SqlCommandFactory.ExecuteCommand( db, tx, sql );
LogMessage("Info", string.Format("Remove {0} records from container:{1} entity:{2}", recordsAffected, ce.ContainerName,
ce.EntityName ));
}
catch(Exception ex)
{
LogMessage("Error", ex);
}
finally
{
tx.Commit();
}


}

const string purgeQuery = @"
DELETE
```

```
public List<Data> Data { get; set; }

}
```

If there are two instances of AiL running within a unit, a development vs. production, the export import functions can act as a way of transferring the code between each system.

Use of Projects to Support Multiple Environments



Name: Production

Server: http://localhost:8000-Oracle

DLL Library Folder:
C:\AiL\EditorDLL
C:\AiL\EditorDLL\ef

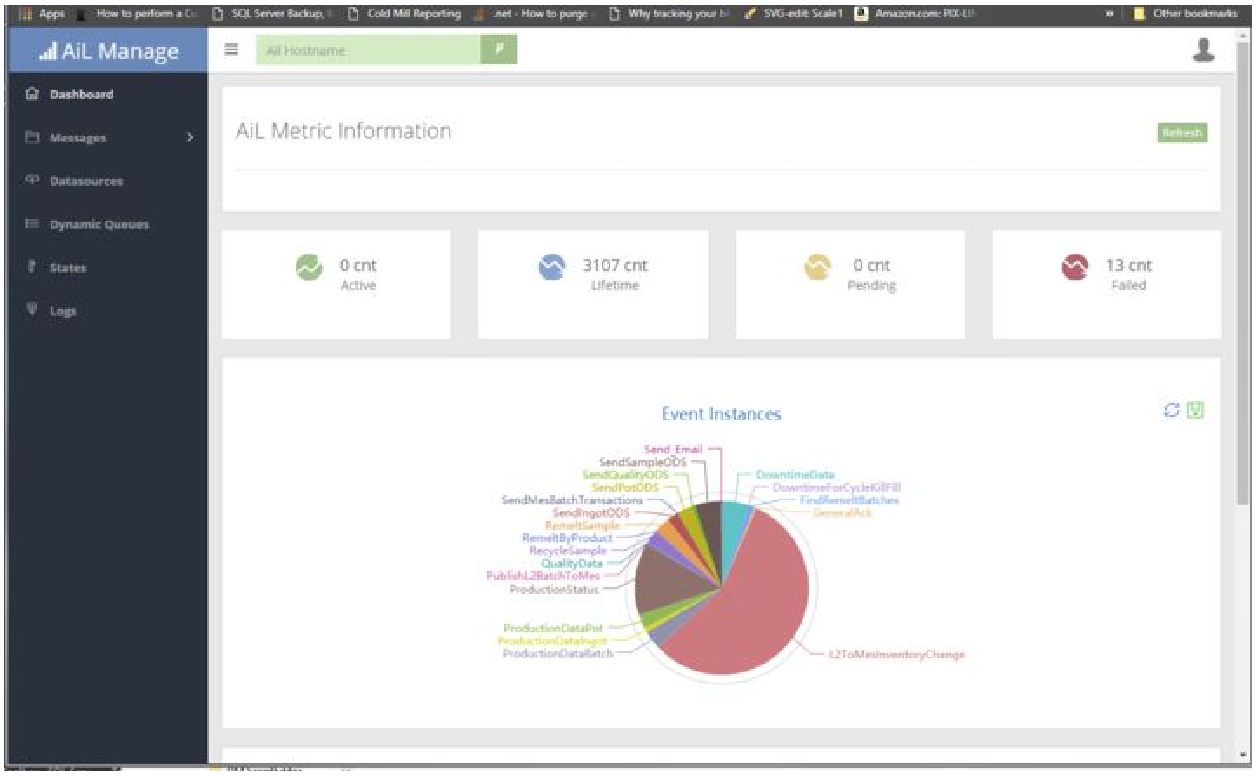
Namespaces:
System.Linq
System.Windows.Forms
System.Data.Common
System.Text.RegularExpressions

Default Client Queueing Library: DM.Library.MessageQ.MsMqClient,DM.Library

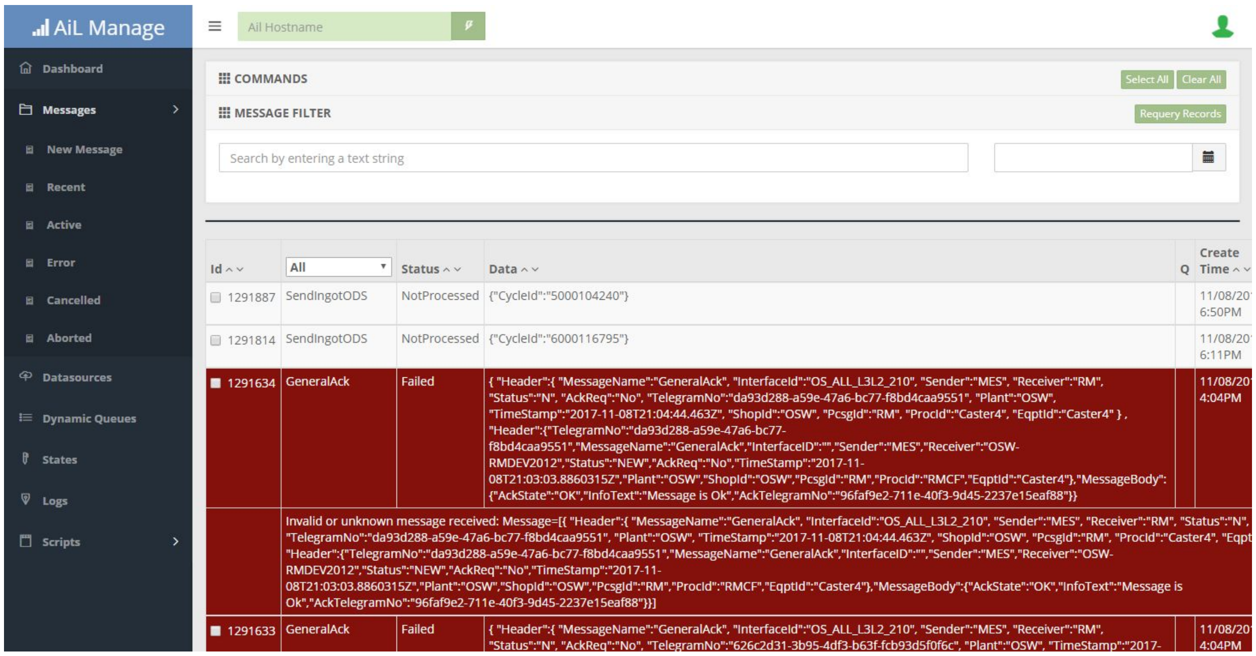
When more than one AiL instance is deployed, in the case of a development environment vs production, use of projects is recommended. Using of projects allows setting up definitions that are unique to each environment such as database connection strings, external DLL's, Queues and Namespaces..

AiL Dashboard Administrator Interface

The AiL dashboard administrator interface is used to manage the AiL environment. This is a web-based tool that can control the AiL service process.

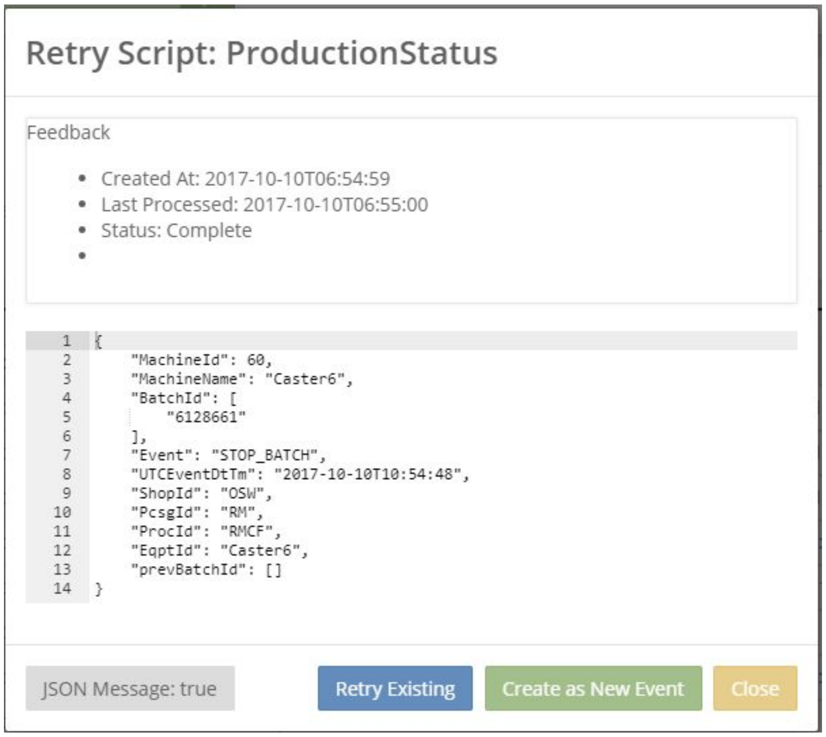


The primary screen provides an overview of event messages that have been or currently being processed.



Id	All	Status	Data	Q	Create Time
1291887	SendIngotODS	NotProcessed	{"CycleId":"5000104240"}		11/08/20 6:50PM
1291814	SendIngotODS	NotProcessed	{"CycleId":"6000116795"}		11/08/20 6:11PM
1291634	GeneralAck	Failed	{ "Header":{ "MessageName":"GeneralAck", "InterfaceId":"OS_ALL_L3L2_210", "Sender":"MES", "Receiver":"RM", "Status":"N", "AckReq":"No", "TelegramNo":"da93d288-a59e-47a6-bc77-f8bd4caa9551", "Plant":"OSW", "TimeStamp":"2017-11-08T21:04:44.463Z", "ShopId":"OSW", "PcsgId":"RM", "ProcId":"Caster4", "EqptId":"Caster4" } , "Header":{"TelegramNo":"da93d288-a59e-47a6-bc77-f8bd4caa9551","MessageName":"GeneralAck","InterfaceID":"","Sender":"MES","Receiver":"OSW-RMDEV2012","Status":"NEW","AckReq":"No","TimeStamp":"2017-11-08T21:03:03.8860315Z","Plant":"OSW","ShopId":"OSW","PcsgId":"RM","ProcId":"RMCF","EqptId":"Caster4"},"MessageBody":{"AckState":"OK","InfoText":"Message is Ok","AckTelegramNo":"96faf9e2-711e-40f3-9d45-2237e15eaf88"}}		11/08/20 4:04PM
	Invalid or unknown message received: Message=[{ "Header":{ "MessageName":"GeneralAck", "InterfaceId":"OS_ALL_L3L2_210", "Sender":"MES", "Receiver":"RM", "Status":"N", "TelegramNo":"da93d288-a59e-47a6-bc77-f8bd4caa9551", "Plant":"OSW", "TimeStamp":"2017-11-08T21:04:44.463Z", "ShopId":"OSW", "PcsgId":"RM", "ProcId":"Caster4", "Eqpt "Header":{"TelegramNo":"da93d288-a59e-47a6-bc77-f8bd4caa9551","MessageName":"GeneralAck","InterfaceID":"","Sender":"MES","Receiver":"OSW-RMDEV2012","Status":"NEW","AckReq":"No","TimeStamp":"2017-11-08T21:03:03.8860315Z","Plant":"OSW","ShopId":"OSW","PcsgId":"RM","ProcId":"RMCF","EqptId":"Caster4"},"MessageBody":{"AckState":"OK","InfoText":"Message is Ok","AckTelegramNo":"96faf9e2-711e-40f3-9d45-2237e15eaf88"}}]				
1291633	GeneralAck	Failed	{ "Header":{ "MessageName":"GeneralAck", "InterfaceId":"OS_ALL_L3L2_210", "Sender":"MES", "Receiver":"RM", "Status":"N", "AckReq":"No", "TelegramNo":"626c2d31-3b95-4df3-b63f-fcb93d5f0f6c", "Plant":"OSW", "TimeStamp":"2017-		11/08/20 4:04PM

Past or current messages can be viewed. Details about the message such as the data payload, execution queue and times are made available for troubleshooting.



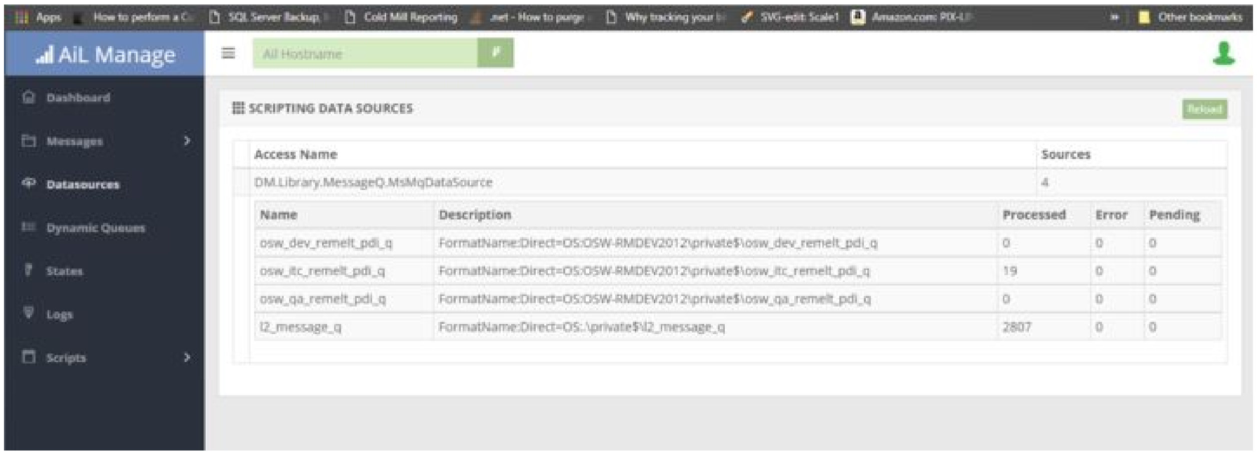
Retry Script: ProductionStatus

Feedback

- Created At: 2017-10-10T06:54:59
- Last Processed: 2017-10-10T06:55:00
- Status: Complete
-

```
{
    "MachineId": 60,
    "MachineName": "Caster6",
    "BatchId": [
        "6128661"
    ],
    "Event": "STOP_BATCH",
    "UTCEventDtTm": "2017-10-10T10:54:48",
    "ShopId": "OSW",
    "PcsgId": "RM",
    "ProcId": "RMCF",
    "EqptId": "Caster6",
    "prevBatchId": []
}
```

Message events can be manipulated by marking them for reprocessing or recreation within the system. If the message contains JSON payload, it will be detected and reformatted for a humanized view of the data.

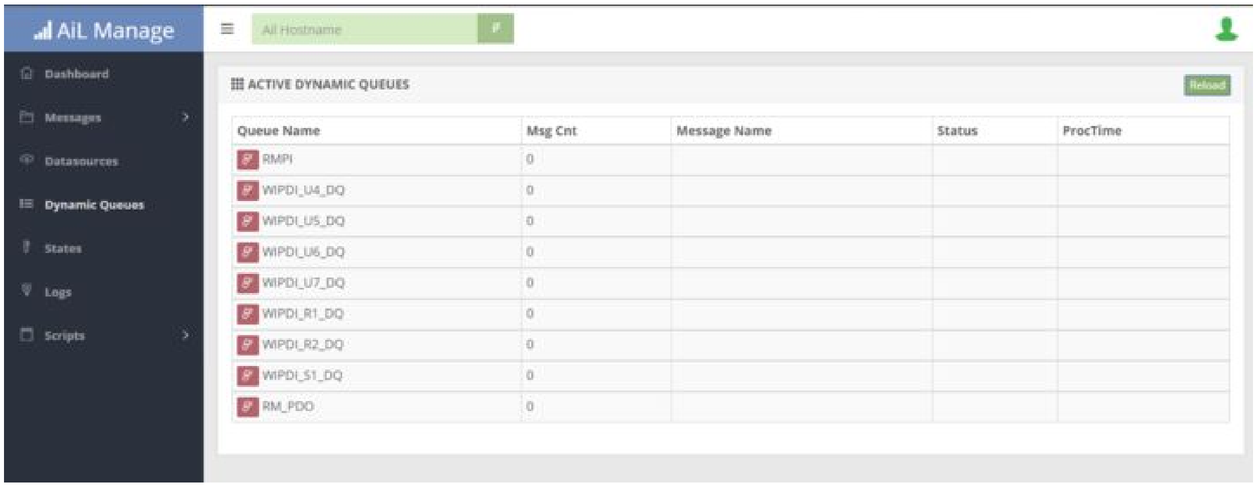


SCRIPTING DATA SOURCES

Access Name	Sources
DM.Library.MessageQ.MsMqDataSource	4

Name	Description	Processed	Error	Pending
osw_dev_remelt_pdi_q	FormatName:Direct=OS:OSW-RMDEV2012\private$\osw_dev_remelt_pdi_q	0	0	0
osw_itc_remelt_pdi_q	FormatName:Direct=OS:OSW-RMDEV2012\private$\osw_itc_remelt_pdi_q	19	0	0
osw_qa_remelt_pdi_q	FormatName:Direct=OS:OSW-RMDEV2012\private$\osw_qa_remelt_pdi_q	0	0	0
l2_message_q	FormatName:Direct=OS:.\private$\l2_message_q	2807	0	0

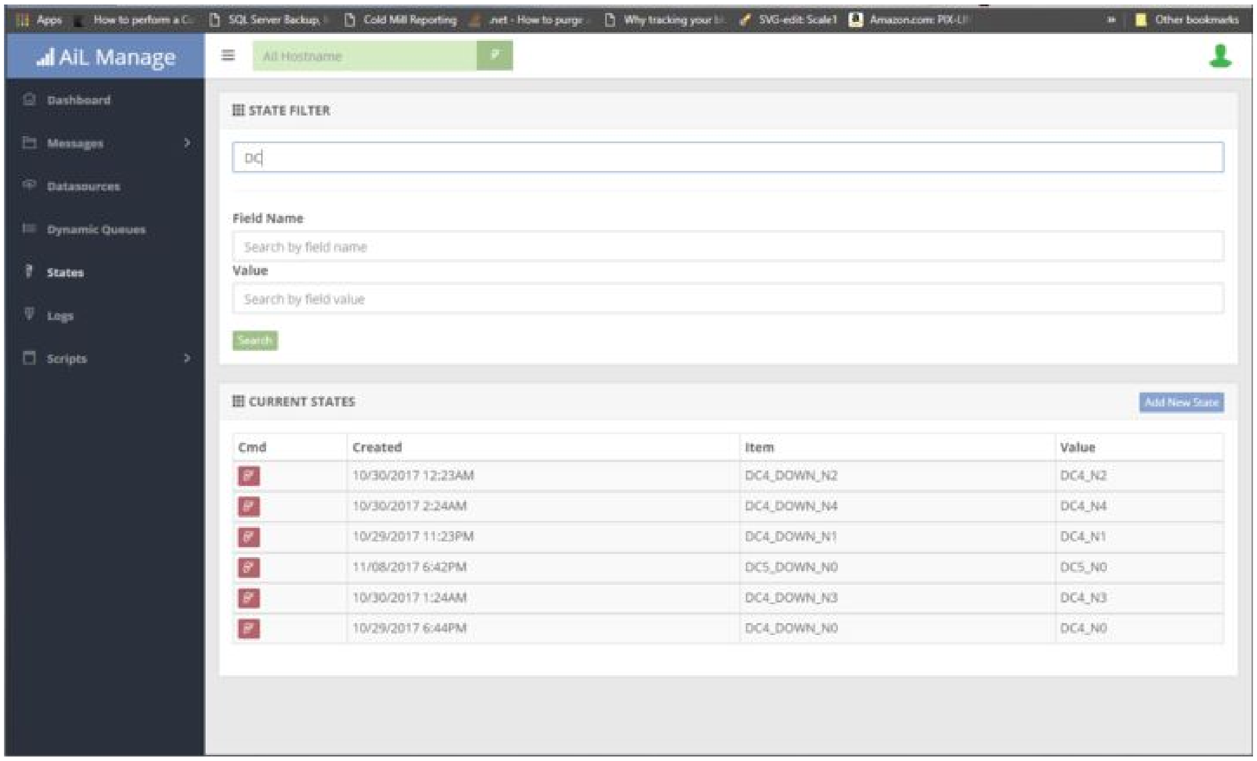
Use the dashboard datasources view to see existing data sources and associated messages that have been processed. Messages that have been completed will be show as Processed. Messages that could not be brought into the AiL environment will be marked as Error. Messages to be processed will be indicated as Pending. Message that are in a Pending state may indicate an broken queue connection.



ACTIVE DYNAMIC QUEUES

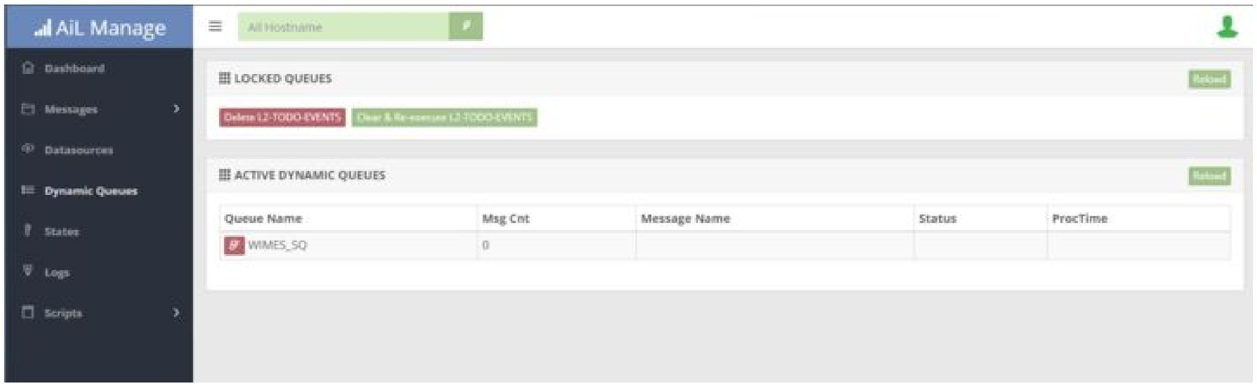
Queue Name	Msg Cnt	Message Name	Status	ProcTime
RMPI	0			
WIPDI_U4_DQ	0			
WIPDI_U5_DQ	0			
WIPDI_U6_DQ	0			
WIPDI_U7_DQ	0			
WIPDI_R1_DQ	0			
WIPDI_R2_DQ	0			
WIPDI_S1_DQ	0			
RM_PDO	0			

View active dynamic queues currently executing in the system. This screen shows 9 active queues with no messages. Dynamic named queues will linger for 2 minutes waiting for new messages before shutting down. Dynamic named queues are automatically started or restarted whenever a message is tagged as such.



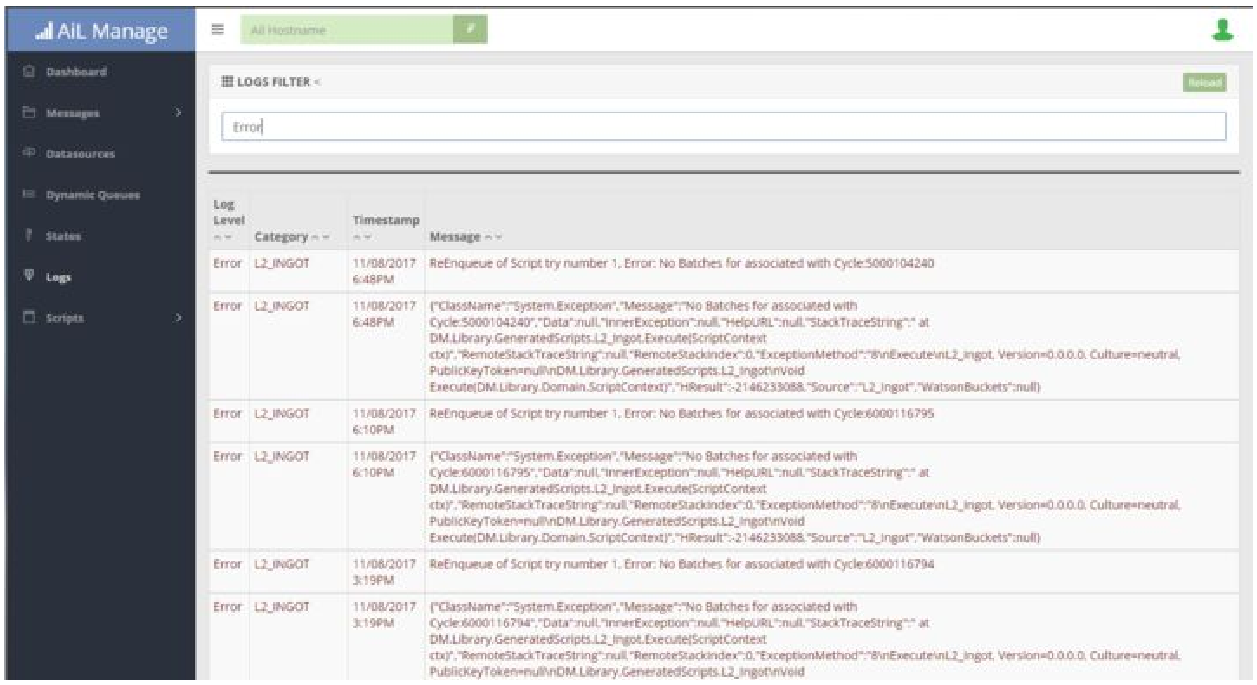
Cmd	Created	Item	Value
🟥	10/30/2017 12:23AM	DC4_DOWN_N2	DC4_N2
🟥	10/30/2017 2:24AM	DC4_DOWN_N4	DC4_N4
🟥	10/29/2017 11:23PM	DC4_DOWN_N1	DC4_N1
🟥	11/08/2017 6:42PM	DC5_DOWN_N0	DC5_N0
🟥	10/30/2017 1:24AM	DC4_DOWN_N3	DC4_N3
🟥	10/29/2017 6:44PM	DC4_DOWN_N0	DC4_N0

States can be used as a form of inter-message communications between script executions. Depending on conditions of data, a state may be used to block another message/script from executing until the data condition is satisfied. Manipulation of states within the script can achieve this function.



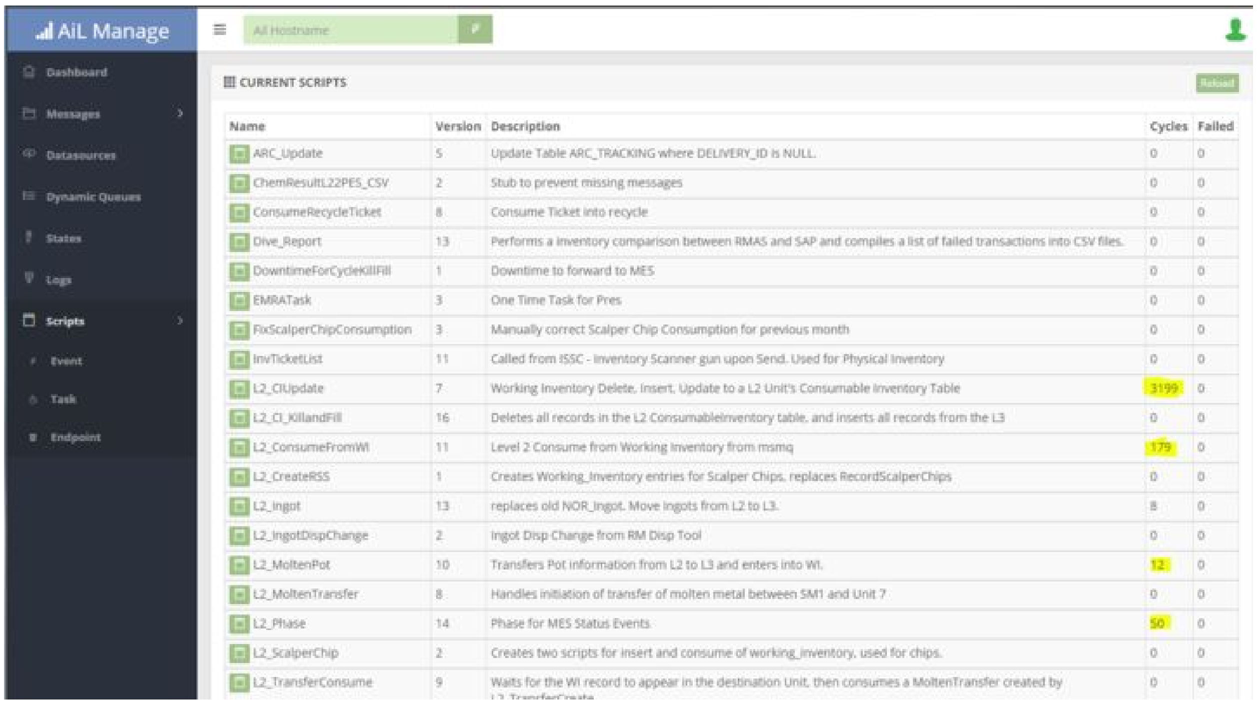
Queue Name	Msg Cnt	Message Name	Status	ProcTime
🟥 WIMES_SQ	0			

Whenever a NQ or DNQ has experienced an unexpected error, the script author may choose to lock the queue. Locked queues will continue to collect messages. Two options exist to clear the lock; clear and reprocess the messages or delete all messages and clear the lock.



Log Level	Category	Timestamp	Message
Error	L2_INGOT	11/08/2017 6:48PM	ReEnqueue of Script try number 1. Error: No Batches for associated with Cycle:5000104240
Error	L2_INGOT	11/08/2017 6:48PM	{"ClassName":"System.Exception","Message":"No Batches for associated with Cycle:5000104240","Data":null,"InnerException":null,"HelpURL":null,"StackTraceString":" at DM.Library.GeneratedScripts.L2_Ingot.Execute(ScriptContext ctx)","RemoteStackTraceString":null,"RemoteStackIndex":0,"ExceptionMethod":"8\nExecute\nL2_Ingot, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null\nDM.Library.GeneratedScripts.L2_Ingot\nVoid Execute(DM.Library.Domain.ScriptContext)","HResult":-2146233088,"Source":"L2_Ingot","WatsonBuckets":null}
Error	L2_INGOT	11/08/2017 6:10PM	ReEnqueue of Script try number 1. Error: No Batches for associated with Cycle:6000116795
Error	L2_INGOT	11/08/2017 6:10PM	{"ClassName":"System.Exception","Message":"No Batches for associated with Cycle:6000116795","Data":null,"InnerException":null,"HelpURL":null,"StackTraceString":" at DM.Library.GeneratedScripts.L2_Ingot.Execute(ScriptContext ctx)","RemoteStackTraceString":null,"RemoteStackIndex":0,"ExceptionMethod":"8\nExecute\nL2_Ingot, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null\nDM.Library.GeneratedScripts.L2_Ingot\nVoid Execute(DM.Library.Domain.ScriptContext)","HResult":-2146233088,"Source":"L2_Ingot","WatsonBuckets":null}
Error	L2_INGOT	11/08/2017 3:19PM	ReEnqueue of Script try number 1. Error: No Batches for associated with Cycle:6000116794
Error	L2_INGOT	11/08/2017 3:19PM	{"ClassName":"System.Exception","Message":"No Batches for associated with Cycle:6000116794","Data":null,"InnerException":null,"HelpURL":null,"StackTraceString":" at DM.Library.GeneratedScripts.L2_Ingot.Execute(ScriptContext ctx)","RemoteStackTraceString":null,"RemoteStackIndex":0,"ExceptionMethod":"8\nExecute\nL2_Ingot, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null\nDM.Library.GeneratedScripts.L2_Ingot\nVoid

Use the dashboard log filter to look for error conditions or feedback. Creating log events within a script can be found in the dashboard log filter provided that the log level is Info or above.



Name	Version	Description	Cycles	Failed
ARC_Update	5	Update Table ARC_TRACKING where DELIVERY_ID is NULL.	0	0
ChemResultL22PES_CSV	2	Stub to prevent missing messages	0	0
ConsumeRecycleTicket	8	Consume Ticket into recycle	0	0
Dive_Report	13	Performs a inventory comparison between RMAS and SAP and compiles a list of failed transactions into CSV files.	0	0
DowntimeForCycleKillFill	1	Downtime to forward to MES	0	0
EMRATask	3	One Time Task for Pres	0	0
FixScalperChipConsumption	3	Manually correct Scalper Chip Consumption for previous month	0	0
InvTicketList	11	Called from ISSC - Inventory Scanner gun upon Send. Used for Physical Inventory	0	0
L2_CIUpdate	7	Working Inventory Delete, Insert, Update to a L2 Unit's Consumable Inventory Table	3199	0
L2_CI_KillandFill	16	Deletes all records in the L2 ConsumableInventory table, and inserts all records from the L3	0	0
L2_ConsumeFromWI	11	Level 2 Consume from Working Inventory from msmq	179	0
L2_CreateRSS	1	Creates Working_Inventory entries for Scalper Chips, replaces RecordScalperChips	0	0
L2_Ingot	13	replaces old NOR_Ingot. Move Ingots from L2 to L3.	8	0
L2_IngotDispChange	2	Ingot Disp Change from RM Disp Tool	0	0
L2_MoltenPot	10	Transfers Pot information from L2 to L3 and enters into WI.	12	0
L2_MoltenTransfer	8	Handles initiation of transfer of molten metal between SM1 and Unit 7	0	0
L2_Phase	14	Phase for MES Status Events	50	0
L2_ScalperChip	2	Creates two scripts for insert and consume of working_inventory, used for chips.	0	0
L2_TransferConsume	9	Waits for the WI record to appear in the destination Unit, then consumes a MoltenTransfer created by	0	0

The dashboard script viewer provides a way to see the active version of the scripts within the service. All scripts are metered by number of executions and CPU times.

Installation

Current deployments of AIL scripting engine is available in either 32 bit or 64 bit distributions. External DLLs requirements will dictate the distribution selection.

Deployments can be obtained from:
https://drive.google.com/drive/folders/0Bwr6nNgfofksUGVob0huMUMxQzQ?usp=sharing

1. Choose the following zip file required for your development environment

Filename	.NET Framework	OS Type
ail-x86-net40-{version}.zip	.NET 4.0	x86
ail-x64-net40-{version}.zip	.NET 4.0	x64
ail-x86-net46-{version}.zip	.NET 4.6	x86
ail-x64-net46-{version}.zip	.NET 4.6	x64

Currently no windows installer is available for AiL environment. The default packages above are configured to use SQL Compact Edition within the connection string. Using the compact edition will allow developers to become familiar with all the features without having to configure an Oracle, SQL Server or Postgres server.

Compact edition can be found at the following url from Microsoft:
https://www.microsoft.com/en-us/download/details.aspx?id=17876

2. Unzip the contents into C:\Ail. The default directory structure for AiL is as follows:

c:\Ail	-> Root directory
c:\Ail\Editor	-> Script editor UI
c:\Ail\EditorDLL	-> Script editor dynamic DLL load folder
c:\Ail\QueueManager	-> AiL Queue Management internal dynamic queue manager UI
c:\Ail\Service	-> Service script executor process
c:\Ail\ServiceDLL	-> Service dynamic DLL load folder

c:\Ail\AilDb.sdf is the compact edition database;

3. Check the application configuration folder to ensure the path names are correct. This will be required if changing the root directory for the application.

In the **<connectionStrings>** section of DM.Service.exe.config and DM.ScriptEditor.exe.config there must be a connection string named “**AiDB”:**
Example connection strings are as follows:

Database Type	Connection String
SQL Compact Edition	<add name="AiDb" connectionString="Data Source=**C:\Ail\AilDb.sdf**;Max Database Size=4000;Persist Security Info=False;" providerName="System.Data.SqlServerCe.4.0"/>
SQL Server	<add name="AiDb" connectionString="Data Source=(localdb)\ProjectsV13;Initial Catalog=AiLDb;Integrated Security=true;" providerName="System.Data.SqlClient"/>
Oracle	<add name="AiDb" connectionString="Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=osw-rmoradev)(PORT=1523))(CONNECT_DATA=(SID=**MYSID**)));User Id=**MYID**;Password=**MYPASS**;" providerName="Oracle.DataAccess.Client"/>

Configuration

Server

Location	Parameter	Description
AppSettings	SchemaUser	Directs the schema used to hold the AIL database schema. If unset, the schema user will default to dbo.
AppSettings	AutoQueueTimeout	Number of minutes of idle activity before terminating a dynamic queue. If no message arrives after a period of time, the queue will be torn down.
AppSettings	MemoryLogLevel	Defines what log level to bubble messages up. The default is 'Info'
AppSettings	MessageRecoveryFolder	Working folder that is used to cache messages that cannot be inserted into the Dispatch Message table.
AppSettings	ShutdownGraceTime	Amount of time to allow for executing messages to complete before terminating the application
AppSettings	QueueLibrary	Default client side queue library to use in the scripting context. Examples: DM.Library.MessageQ.FileWriter,DM.Library DM.Library.MessageQ.MsMqClient,DM.Library Amq.Messaging.AmqWriter,Amq.Messaging
AppSettings	TransactionLogFolder	(Future) Global transaction log file to track incoming and outgoing messages; used as a json document; to be presented at the web layer
AppSettings	HttpEndpointBasicAuthType	This parameter determines what authorization store to use when authenticated endpoint requests that are enabled with HTTP-AUTH. Active directory authentication always will rely on the current active directory environment that the process is running in to get its configuration. **Active Directory Usage:** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.ActiveDirectory,DM.Library"/> **Always Fail Auth** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.AlwaysFail,DM.Library"/> **Always Auth** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.AlwaysAuthorize,DM.Library"/>

		Simple File <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.SimpleFile,DM.Library"/> Simple file authentication requires userlist.json to be present in the working folder userlist.json should have
AppSettings	AssemblyReferences	Allow dynamically call for individual .NET DLL files comma delimited.
AppSettings	AssemblyReferenceFolders	AssemblyReferenceFolders allow dynamically call for folders in which all .NET DLLs in the folder are loaded
AppSettings	ScriptNamespaces	Scripting extensions; allow namespaces to be injected into dispatch manager scripts.
AppSettings	DaysToKeepMessages	Number of days to retain Processed and Failed messages.
AppSettings	DaysToKeepStates	Number of days to retain State
AppSettings	MailServer	SMTP Server used to send email notifications when there is a system failure.
AppSettings	MailFrom	SMTP Mail From Server
AppSettings	MailServerPort	Mail server port: 25 or 587 for secure mail
AppSettings	MailUsername	Username sending the email
AppSettings	MailPassword	Password used for sending the email
AppSettings	MailSendTo	Default email addresses to send the email
AppSettings	TimeBetweenFailures	When there is more than one failure of the same message type, this setting prevent multiple messages from appearing with continuous failures. The setting is in minutes .
AppSettings	AppSyncName	Used when more than one AiL server will be running on the same machine.
Scripting QueueConnections	Section Configuration	This section is used to define queues that will be read or written from within the application. Refer to Queue configuration to examine differences that may exist between Active MQ, RabbitMq and Microsoft Message Q.
Scripting Registered DataSources	Section Configuration	Registered data sources allow a way to a permanent tasks started within the AiL system. Preconfigured data sources available is the ActiveMq and Microsoft Message Q data source. These data sources are used to accept messages from inbound queues and place them in the Dispatch Message event table.

Client

Client app.config settings are generally a subset of the server settings.

Location	Parameter	Description
AppSettings	SchemaUser	Directs the schema used to hold the AIL database schema. If unset, the schema user will default to dbo.
AppSettings	DispatchManagerAppServer	This is the connection path used to communicate directly with a target DM.Service. This url is used to post script updates and reloads from within the editor.
AppSettings	QueueLibrary	Default client side queue library to use in the scripting context. Examples: DM.Library.MessageQ.FileWriter,DM.Library DM.Library.MessageQ.MsMqClient,DM.Library Amq.Messaging.AmqWriter,Amq.Messaging
AppSettings	TransactionLogFolder	(Future) Global transaction log file to track incoming and outgoing messages; used as a json document; to be presented at the web layer
AppSettings	HttpEndpointBasicAuthType	This parameter determines what authorization store to use when authenticated endpoint requests that are enabled with HTTP-AUTH. Active directory authentication always will rely on the current active directory environment that the process is running in to get its configuration. **Active Directory Usage:** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.ActiveDirectory,DM.Library"/> **Always Fail Auth** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.AlwaysFail,DM.Library"/> **Always Auth** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.AlwaysAuthorize,DM.Library"/> **Simple File** <add key="HttpEndpointBasicAuthType" value="DM.Library.Auth.SimpleFile,DM.Library"/> Simple file authentication requires userlist.json to be present in the working folder userlist.json should have
AppSettings	AssemblyReferences	Allow dynamically call for individual .NET DLL files comma

		delimited.
AppSettings	AssemblyReferenceFolders	AssemblyReferenceFolders allow dynamically call for folders in which all .NET DLLs in the folder are loaded
AppSettings	ScriptNamespaces	Scripting extensions; allow namespaces to be injected into dispatch manager scripts.
Scripting QueueConnections	Section Configuration	This section is used to define queues that will be read or written from within the application. Refer to Queue configuration to examine differences that may exist between Active MQ, RabbitMq and Microsoft Message Q.

Script Coding (C#) Examples

1. Start the Script Editor IDE.
2. Select "New Script" from the Script option on the Menu bar.
3. Enter the relevant information and settings and click OK.

Begin by entering code into the Execute Method.

```
public void Execute(DM.Library.Domain.ScriptContext ctx)
{
        //your code here
}
```

To make requests to a database defined in a connection string:

```
        using (var db = SqlCommandFactory.GetDatabase("AiDb"))
        {
            var cmd = db.CreateCommand();

            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM TEST_TABLE where id in (1,3)";
            cmd.CommandTimeout = 60t;

            Var result = cmd.ExecuteScalar();
        }
```

To send a message using MSMQ:

```
        ctx.Queue.Enqueue("Test_Inbound_Queue", "",
        "TestEvent?TargetQueueName=MYQUEUE_1234&Timeout=100", "");
```

To insert an entry into the log:

```
        LogMessage("Info", "Your Log Entry Here");
```

Components

Data Connection Strings

Examples:

Oracle:

```
	<add name="AiDb" connectionString="Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=Test)(PORT=1523))(CONNECT_DATA=(SID=Test)));U
ser Id=user;Password=123;" providerName="Oracle.DataAccess.Client"/>
```

Amazon Redshift:

```
<add name="Amazon" connectionString="Driver={Amazon Redshift (x86)};
Server=myserver.redshift.amazonaws.com; Database=Test; UID=user; PWD=123; Port=5439"
providerName="System.Data.Odbc"/>
```

SQL Server Compact (CE):

```
	<add name="AiDb" connectionString="Data Source=..\AilDb.sdf;Max Database
Size=4000;Persist Security Info=False;" providerName="System.Data.SqlServerCe.4.0"/>
```

SQL Factory Data Library

Built into the script engine is a standard way to work with ADO.NET accessible databases. The SQL Factory allows connections to external databases for SQL command execution.

Scripting Registered Data Sources

Registered data sources provide a way to integrate inbound data messages from different sources. A data source can be a file transfer, file watch or queuing event (Microsoft Message Queue) which will map to an event script.

New data sources can be created by writing to the ***IDataSource*** interface within the DM.Library module.

Scripting Queue Connections

Scripting queue connections provide a way to create queue events and react to inbound queue events in a generalized fashion.

Examples:

Microsoft Message Queuing (MSMQ):

```
<Queue name="Test_Inbound_Queue"
       clientClass="DM.Library.MessageQ.MsMqClient,DM.Library"
       serverClass="DM.Library.MessageQ.MsMqDataSource,DM.Library">
  <Parameters>
    <Parameter name="queueName" value="test_message_q"/>
    <Parameter name="queuePath" value="FormatName:Direct=OS:.\dir\test_message_q"/>
    <Parameter name="formatterType" value="Xml"/>

    <!-- Possible values that would define a message location
            msmq:label = default if blank
            msmq:defaultMessage = use defaultMessage from the file
            json:<path>
            future:  xml:<path>
      -->
    <Parameter name="messageTypePath" value="json:MessageName"/>
    <Parameter name="defaultMessage" value="test_Unknown_Message"/>
  </Parameters>
</Queue>
```