External Messaging Agent

Advanced Design and Setup Guide

Version: 7.3.x

Written by: Product Knowledge, R&D
Date: June 2019

Hyland

Table of Contents

About External Messaging Agent

The Perceptive Content External Messaging Agent (inserverEM.exe) allows external systems to trigger events in Perceptive Content. External Messaging Agent is included as part of the Perceptive Content Server installation and is responsible for processing external messages inserted directly into Perceptive Content external message tables. Because the External Messaging Agent is an asynchronous process, it is different from general web services solutions such as Message Agent and Envoy.

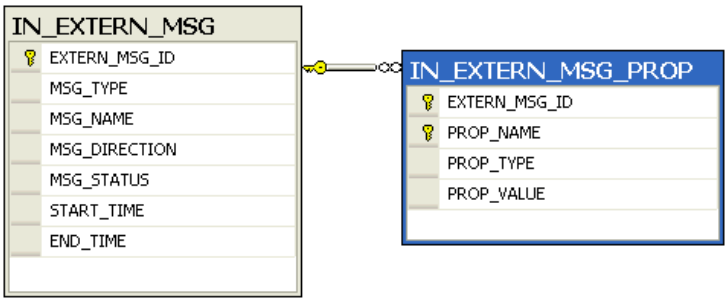
When to use External Messaging Agent

You should consider using External Messaging Agent if the following scenarios are true in your situation:

- Your system has SQL capabilities and is integrated with Perceptive Content.
- Your system needs to signal some type of action to be taken by custom iScripts.
- You need to employ event driven rather than scheduled tasks.
- You need increased reliability and logging capabilities, such as for HL7 compliance.
- You prefer simplified iScript strategy that enables data to be passed directly to the iScript without the need for the iScript to connect to a database or parse a flat file.
- You need to increase efficiency by minimizing database polling instances.

How External Messaging Agent works

Using INEMUSER, your external system connects to the Perceptive Content database via JOBC or ODBC and inserts rows in the external message tables. Data passes to the iScript using name-value pairs and the IN_EXTERN_MSG_PROP table, as demonstrated in the following example:



Case study

The following case study demonstrates how you set up an external system to use External Messaging Agent. Suppose your company's Human Resources department uses PeopleSoft as the host system for Perceptive Content. The HR administrator needs to change Sally Smith's last name to Johnson without creating a new employee ID for her and maintaining her employee ID of 123456. To complete this task, the administrator completes the name change in PeopleSoft, which activates a name-change script that uses SQL to insert rows in your Perceptive Content database.

During implementation of Perceptive Content, the External Message Agent configuration file, inserverEM.ini, was configured to execute a custom iScript, HR_UpdateName.js, for an employee name change. As a result of the host system inserting this message, INEMUSER runs the iScript, which updates all of Sally's documents by changing her last name to Johnson while maintaining her employee ID.

In this case study, the following message is inserted into the agent:

```
Begin Trans
insert into in_extern_msg (EXTERN_MSG_ID, MSG_TYPE, MSG_NAME, MSG_DIRECTION,
MSG_STATUS, START_TIME) values ( 'PS_1227306536156', 'Employee_Update', 'Update_Name',
1, 1, CURRENT_TIMESTAMP)
insert into in_extern_msg_prop (EXTERN_MSG_ID, PROP_NAME, PROP_TYPE, PROP_VALUE)
values ('PS_1227306536156', 'emplId', 0,'12356')
insert into in_extern_msg_prop (EXTERN_MSG_ID, PROP_NAME, PROP_TYPE, PROP_VALUE)
values ('PS_1227306536156', 'newLastName', 0,'Smith')
Commit Trans
```

To process this message, inserverEM.ini contains the following configuration:

```
[General]
num.workers=5
message.pulling.interval=10
max.messages.per.type=100

[Employee_Update]
Update_Name=HR_UpdateName.js
Update_EmplId=HR_UpdateEmplId.js

[Some_other_Process]
PurgeDocument=HR_PurgeDocument.js
```

When received, inserverEM.ini processes the message and executes HR_UpdateName.js, which contains the following code:

```
function main()
{
	var externalMsgObj = getInputPairs();
	var emplId = externalMsgObj["emplId"];
	var newName = externalMsgObj["newLastName"];
	printf("Updating employee %s with new last name of %s\n", emplId, newName);
```

iScript considerations

When creating custom scripts for External Messaging Agent, consider the following points:

- The Windows user running External Messaging Agent runs the iScript, which must be stored in [*drive:*]\inserver\script.
- The iScript function, getInputPairs()returns a object (or an associated array/hash) with the name-value pairs of the data in the IN_EXTERN_MSG_PROP table as well as the data in the IN_EXTERN_MSG row.
- If the script calls setSuccess(false), then the script failed to process the message. You can create a scheduled task or SQL script to set the msg_status from 4 to 1 to initiate a retry.

Additional considerations

As you plan your External Messaging Agent implementation, consider the following points:

- Your external system is responsible for generating a unique ID that is used to link the properties to the message.
- You must use database transactions to perform table inserts so you can roll back a transaction if an error occurs.
- You must ensure that the external system properly disconnects from the database and does not consume resources, which could negatively affect the performance of Perceptive Content.

Appendix A: Table column definitions

IN_EXTERN_MSG

Column	Description
EXTERN_MSG_ID	The unique identifier for the message.
MSG_TYPE	The message type or category as defined in the External Messaging Agent configuration file, inserverEM.ini.
MSG_NAME	The message name as defined in the External Messaging Agent configuration file, inserverEM.ini.
MSG_DIRECTION	The direction of the message. 1 = inbound 0 = outbound
MSG_STATUS	Describes the status of the message. External Messaging Agent updates the status as it processes the message. **0 - UNDEFINED** Indicates the message is new. This status prevents messages from processing before properties are set. **1 - NEW** The message is ready to be processed by the agent. **2 - BEING_PROCESSED** The agent has stored the message into memory and is processing it. **3 - COMPLETE** The script successfully processed the message. **4 - COMPLETE_WITH_ERROR** The script failed to process the message. **999 - STATUS_LOCKED** The agent is reading the message from the external table. This status occurs before **BEING_PROCESSED**. **Note** The iScript can signal that it was unable to successfully process the message by calling setSuccess(false); However, the script will not be retried again.
START_TIME	The time the external system inserted the message, based on the external system local time.
END_TIME	The time the external system stopped inserting the message, based on the external system local time. This will be updated by the agent and does not need to be populated by the external system.

IN_EXTERN_MSG_PROP

Column	Description
EXTERN_MSG_ID	The unique identifier for the message.
PROP_NAME	The data property name passed to the processing iScript.
PROP_TYPE	The property type, which must be 0. **Note** The IN_EXTERN_MSG_PROP table supports only the string data type, which corresponds to a value of 0 in the PROP_TYPE column.
PROP_VALUE	The data property value to be passed to the processing iScript.