Understanding Plug-ins in MS CRM 2011

A plug-in is custom business logic (code) that you can integrate with Microsoft Dynamics CRM 2011 and Microsoft Dynamics CRM Online to modify or augment the standard behavior of the platform.

Another way to think about plug-ins is that they are handlers for events fired by the Microsoft Dynamics CRM platform. You can subscribe, also known as registering, a plug-in to a known set of events to have your code run when the event occurs.

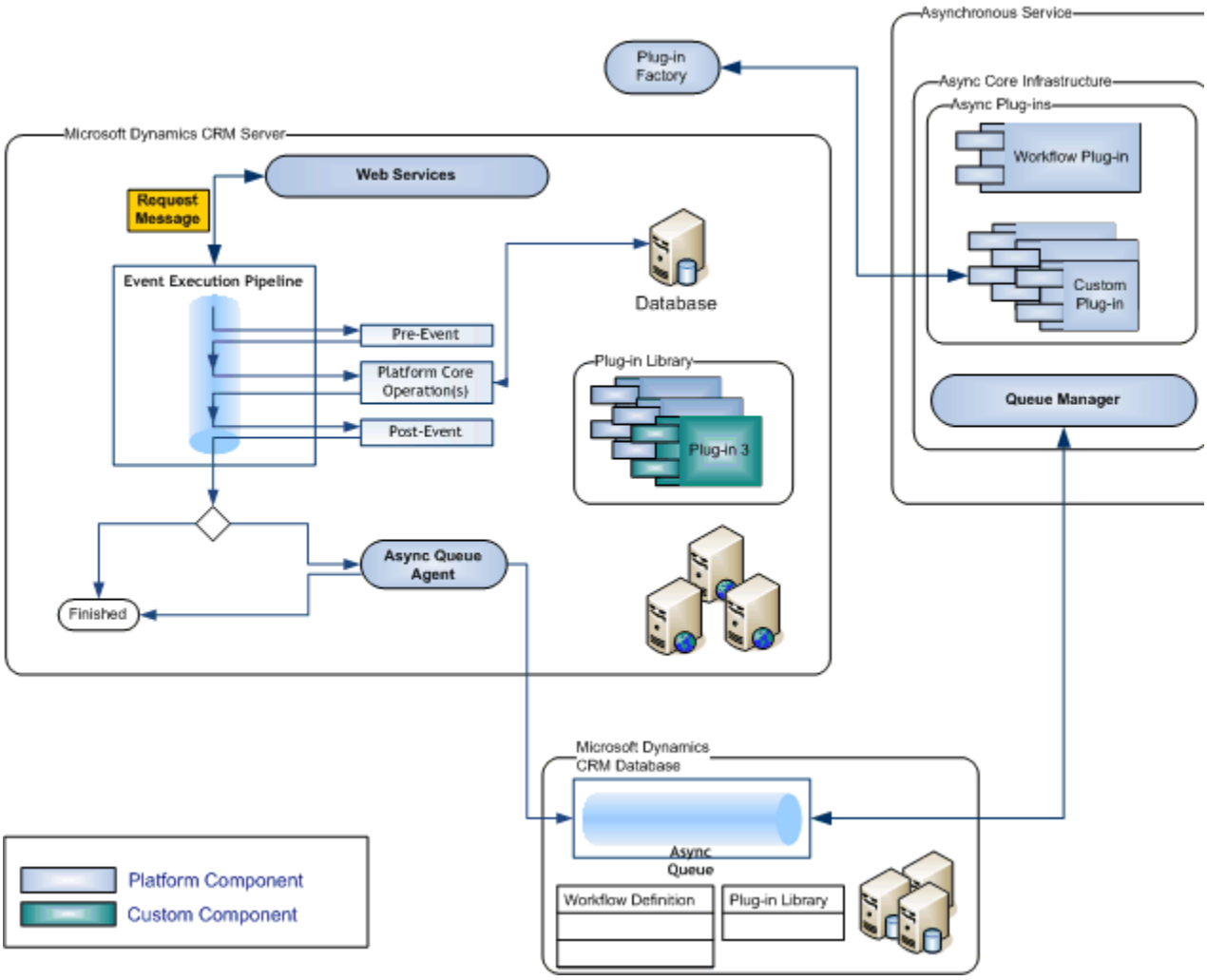
Microsoft Dynamics CRM 2011 - Event Execution Pipeline

The Microsoft Dynamics CRM event processing subsystem executes plug-ins based on a message pipeline execution model. A user action in the Microsoft Dynamics CRM Web application or an SDK method call by a plug-in or other application results in a message being sent to the organization Web service. The message contains business entity information and core operation information. The message is passed through the event execution pipeline where it can be read or modified by the platform core operation and any registered plug-ins.

Note: While there are several Web services hosted by the Microsoft Dynamics CRM platform, only events triggered by the organization and OData endpoints can cause plug-ins to execute.

Architecture and Related Components

The following figure illustrates the overall architecture of the Microsoft Dynamics CRM platform with respect to both synchronous and asynchronous event processing.



Event Processing Architecture

The event execution pipeline processes events either synchronously or asynchronously. The platform core operation and any plug-ins registered for synchronous execution are executed immediately. Synchronous plug-ins that are registered for the event are executed in a well-defined order. Plug-ins registered for asynchronous execution are queued by the Asynchronous Queue Agent and executed at a later time by the asynchronous service.

Important: Regardless of whether a plug-in executes synchronously or asynchronously, there is a 2 minute time limit imposed on the execution of a (message) request. If the execution of your plug-in logic exceeds the time limit, a System.TimeoutException is thrown. If a plug-in needs more processing time than the 2 minute time limit, consider using a workflow or other background process to accomplish the intended task.

Pipeline Stages

The event pipeline is divided into multiple stages, of which 4 are available to register custom developed or 3rd party plug-ins. Multiple plug-ins that are registered in each stage can be further be ordered (ranked) within that stage during plug-in registration.

Event: Pre-Event
Stage name: Pre-validation
Stage number: 10
Description: Stage in the pipeline for plug-ins that are to execute before the main system operation. Plug-ins registered in this stage may execute outside the database transaction. The pre-validation stage occurs prior to security checks being performed to verify the calling or logged on user has the correct permissions to perform the intended operation.

Event: Pre-Event
Stage name: Pre-operation
Stage number: 20
Description: Stage in the pipeline for plug-ins that are to execute before the main system operation. Plug-ins registered in this stage are executed within the database transaction.

Event: Platform Core Operation
Stage name: Main Operation
Stage number: 30
Description: In-transaction main operation of the system, such as create, update, delete, and so on. No custom plug-ins can be registered in this stage. For internal use only.

Event: Post-Event
Stage name: Post-operation
Stage number: 40
Description: Stage in the pipeline for plug-ins which are to execute after the main operation. Plug-ins registered in this stage are executed within the database transaction.

Event: Post-Event
Stage name: Post-operation (Deprecated)
Stage number: 50
Description: Stage in the pipeline for plug-ins which are to execute after the main operation. Plug-ins registered in this stage may execute outside the database transaction. This stage only supports Microsoft Dynamics CRM 4.0 based plug-ins.

Message Processing

Whenever application code or a workflow invokes a Microsoft Dynamics CRM Web service method, a state change in the system occurs that raises an event. The information passed as a parameter to the Web service method is internally packaged up into a OrganizationRequest message and processed by the pipeline. The information in the OrganizationRequest message is passed to the first plug-in registered for that event where it can be read or modified before being passed to the next registered plug-in for that event and so on. Plug-ins receive the message information in the form of context that is passed to their Execute method. The message is also passed to the platform core operation.

Plug-in Registration

Plug-ins can be registered to execute before or after the core platform operation. Pre-event registered plug-ins receive the OrganizationRequest message first and can modify the message information before the message is passed to the core operation. After the core platform operation has completed, the message is then known as the OrganizationResponse. The response is passed to the registered post-event plug-ins. Post-event plug-ins have the opportunity to modify the message before a copy of the response is passed to any registered asynchronous plug-ins. Finally, the response is returned to the application or workflow that invoked the original Web service method call.

Because a single Microsoft Dynamics CRM server can host more than one organization, the execution pipeline is organization specific. There is a virtual pipeline for every organization. Plug-ins registered with the pipeline can only process business data for a single organization. A plug-in that is designed to work with multiple organizations must be registered with each organization's execution pipeline.

Note: In Microsoft Dynamics CRM 4.0, there existed a parent and a child pipeline. These pipelines have been consolidated into one pipeline for this release. A pre-event plug-in registered in the parent pipeline of Microsoft Dynamics CRM 4.0 is equivalent to registering in stage 10. A pre-event plug-in registered in the child pipeline of Microsoft Dynamics CRM 4.0 is equivalent to registering in stage 20.

Inclusion in Database Transactions

Plug-ins may or may not execute within the database transaction of the Microsoft Dynamics CRM platform. Whether a plug-in is part of the transaction is dependent on how the message request is processed by the pipeline. You can check if the plug-in is executing in-transaction by reading the IsInTransaction property inherited by IPluginExecutionContext that is passed to the plug-in. If a plug-in is executing in the database transaction and allows an exception to be passed back to the platform, the entire transaction will be rolled back. Stages 20 and 40 are guaranteed to be part of the database transaction while stage 10 and 50 may be part of the transaction.

Any registered plug-in that executes during the database transaction and that passes an exception back to the platform cancels the core operation. This results in a rollback of the core operation. In addition, any pre-event or post event registered plug-ins that have not yet executed and any workflow that is triggered by the same event that the plug-in was registered for will not execute.

CRM Event Pipeline

- Calls to the CRM web services execute some message.
- Each message creates a pipeline made up of various stages.
- Plug-ins can be registered on different stages in the pipeline
- A CRM “2011” plug-in is a .NET class that implements**Microsoft.Xrm.Sdk.IPlugin**
- Assemblies registered with CRM and stored in database, GAC or on disk

CRM Event Pipeline Visualization

• **PreEvent**

• Stage 10: Pre-validation

• **PreEvent**

• Stage 20: Pre-operation

• **Core Operation**

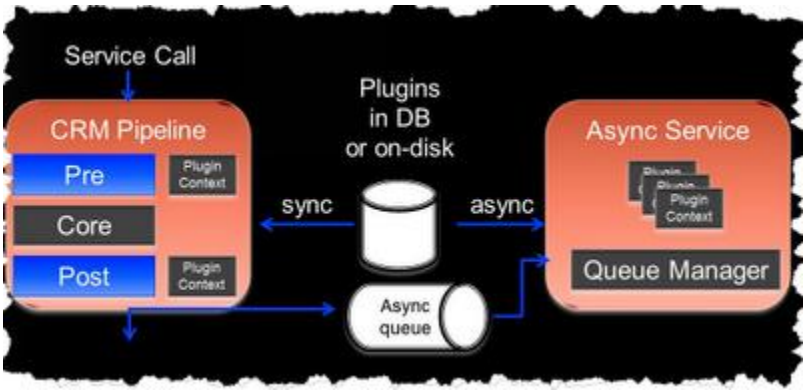
• Stage 30: MainOperation

• **PostEvent**

• Stage 40: Post-operation

• **PostEvent**

• Stage 50: Post-operation (deprecated*)



Note: 20,30,40 are ***Transaction Scope***

Parent/Child Pipelines

- Parent and Child Pipelines no longer exist
- Pre-event Plugin in Parent Pipeline in CRM 4.0

– Pre-validation in CRM 2011 (outside transaction)

- Pre-event Plugin in Child Pipeline in CRM 4.0

– Pre-operation in CRM 2011 (inside transaction)

Sync/Async Execution

- Plugins can execute

- Synchronously during pipeline execution

- Asynchronously – queued for later execution

Plug-ins in CRM4

- Plug-ins ran in the same process context and not isolated
- Plug-ins could not participate in SQL transactions
- Plug-in registration had to be done by a Deployment Administrator

Plug-ins in CRM 2011

- Able to participate in SQL transactions
- Able to create traces returned with exceptions
- Plug-in assemblies can have 2 isolation modes:

- None or Sandbox

- The addition of the Sandbox isolation mode enables the use of plug-ins in CRM Online.

- Custom workflow activities will not be enabled in CRM Online for CRM 2011

Writing a Plugin

- Implement *IPlugin*

– Just the *Execute* method

- Constructor can take two strings

– Unsecure config

– Secure config

- Work with data from *IPluginExecutionContext*

– *Target,* pre-images and post-images etc…

– Handed a late bound *Entity*
Can transform to early bound type using generic method
targetEntity.ToEntity()

- Ensure plugin code is stateless

– Do not assume state is maintained between executions

Pre/Post Event Images

- Snapshot of Entity state at that point in time
- Removes need to call RetrieveRequest to get entity state during execution
- Attributes to be imaged must be specified in registration

- *SdkMessageProcessingStepRegistration.Images*

- Available via properties at runtime

- Pre: *IPluginExecutionContext.PreEntityImages*

- Post: *IPluginExecutionContext.PostEntityImages*

Offline Support

- Offline plugins execute in context of CRM Outlook Add-in
- Plugins can be: Online/Offline/Both
- Check if offline via *%ExecutionContext%.IsExecutingOffline*
- Offline plugins should be idempotent

- May be executed twice

- Use *IsOfflinePlayback* property to check

Transaction Support

- CRM 2011 Stages support plugin execution inside the database transaction

- Stages 20 & 40

- Uncaught exceptions force a rollback
- *IExecutionContext.IsInTransaction*
- Transaction spans CRM DB operations only

- No distributed transaction support

- Plugins registered outside transaction stage **will** participate in transaction if executed as nested pipeline of transactional parent

CRM 2011 – Images and Parameters

1. Input and Output Parameters
2. Pre and Post Images

Input and Output parameters

The event execution pipeline passes the request message as input parameters. The plugin code has access to this data in form of parameter collection, which is essentially name value pairs. Again, note that the parameter collection depends on the type of request. One of the most important thing to note here is that the input parameters only consists of “changed” data. This is a very important which took me a while to learn. So for update request, the input parameters will only have fields which changed, whereas for create, as you can guess, it would have all the properties. Similarly, the event execution pipeline would pass in the response message as output parameters. Similar to input parameters, properties available depend on type of request and the execution pattern (synchronous/asynchronous) for the plugin.

e.g.:

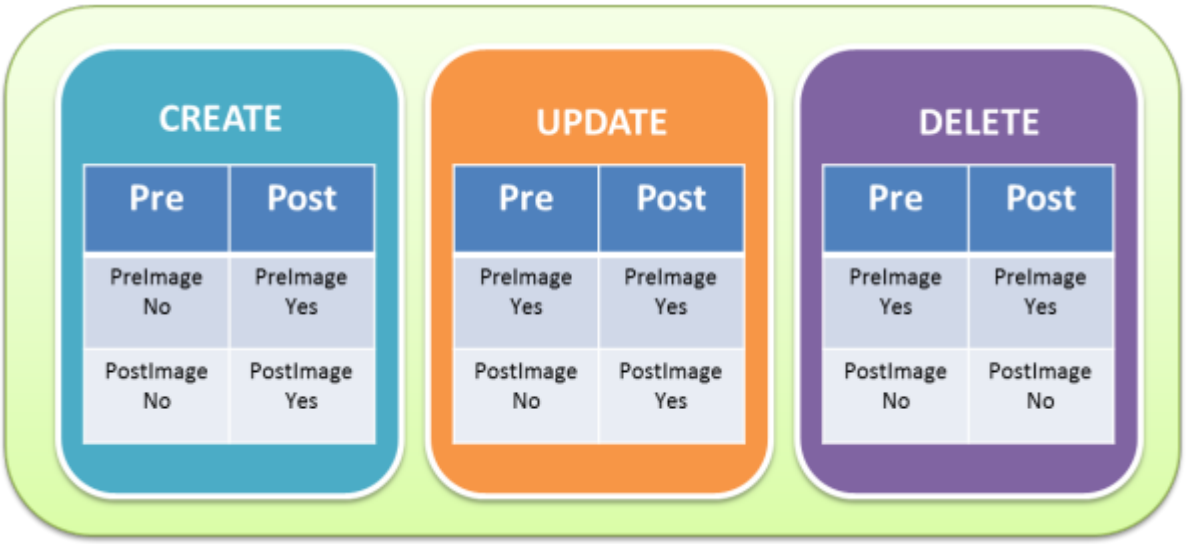
```
Entity callingEntity = (Entity)Context.InputParameters["Target"];

EntityReferenceCollection relatedentities =
(EntityReferenceCollection)context.InputParameters["RelatedEntities"];
```

Pre and Post Images

Pre and Post images contain snapshots of the primary entity’s attributes before and after the core platform operation. Microsoft Dynamics CRM populates the pre-entity and post-entity images based on the security privileges of the impersonated system user. You can specify to have the platform populate these properties when you register your plug-in. The entity alias value you specify during plug-in registration is used as the key into the image collection. Once your plugin and steps are registered, register new image using the plugin registration tool. You have to make sure to use the same name in the EntityAlias field which you will use to query in the plugin code. For

e.g.: PluginExecutionContext.PreEntityImages.Contains(“PreImageXML”), make sure to use “PreImageXML” as EntityAlias while registering image.

You will need to repeat same steps for post image as well. For best performance, only select attributes which are used in the plugin code to be passed in the pre and post images. Passing all attributes would slow down the plugin execution. There are some events where images are not available. Below is a quick summary of pre and postimage support in CRM 2011.


CREATE
Pre
Post
PreImage No
PreImage Yes
PostImage No
PostImage Yes
UPDATE
Pre
Post
PreImage Yes
PreImage Yes
PostImage No
PostImage Yes
DELETE
Pre
Post
PreImage Yes
PreImage Yes
PostImage No
PostImage No