However, only synchronous post-event and asynchronous registered plug-ins have OutputParameters populated as the response is the result of the core platform operation. The property is of type ParameterCollection where the keys to access the response data are the names of the actual public properties in the response.

There are some events where images aren’t available. For example, only synchronous post-event and asynchronous registered plug-ins have PostEntityImages populated. The create operation doesn’t support a pre-image and a delete operation doesn’t support a post-image. In addition, only a small subset of messages support pre and post images as shown in the following table.

You need to consider first the deployment type, if it is for CRM Online or CRM OnPremise.

For CRM Online plugin assemblies must be registered in Sandbox.

For CRM OnPremise plugin assemblies can be registered in Sandbox or outside the Sandbox (option None).

If the isolation mode is Sandbox the location is always Database, the advantage is that plugins stored in the database are automatically distributed across multiple CRM servers in a data center cluster.

the disadvantage is that you can't install external assemblies to the database, so if your plugin uses an external assembly you need to merge it with your plugin assembly before the registration or deploy the external assembly manually to the GAC or the the CRM bin folder.

If your register your plugin assembly outside the Sandbox (meaning you are OnPremise) you can choose to deploy it to the GAC, in this case the advantage is that you take full advantage of the GACs versioning system, preventing conflicting versions of the same assembly if multiple versions are needed. The disadvantage is that the registration requires gacutil.exe and this can be an issue for some deployments.

If you choose to deploy to Disk, the plugin assembly will be copied to the CRM bin folder, in this case the debug will be easier (but you can debug the plugins also when they are registered in the database) but you lose also the GAC versioning advantage.

<https://crmbusiness.wordpress.com/2015/06/29/crm-2015-plugin-deployment-options/>

Sand box Plugins:

<https://crmbusiness.wordpress.com/2015/02/05/understanding-plugin-sandbox-mode/>

<https://crmbusiness.wordpress.com/mb2-703-crm-2013-customization-and-configuration-certification/>

<http://rahuljim.blogspot.in/2013/09/different-way-to-register-your-plugins.html>

<http://gonzaloruizcrm.blogspot.in/2011/06/6-great-reasons-to-register-your.html>

<https://social.microsoft.com/Forums/en-US/4aa2a640-acdc-4a00-a638-d4dbf73af113/crm-2011-plugin-message-create-does-not-support-this-image-type?forum=crmdevelopment>

<https://sliong.wordpress.com/2012/06/06/crm-2011-event-execution-pipeline-and-target-input-parameters/>

important :

<http://stackoverflow.com/questions/23496229/difference-between-pre-operation-and-post-operation-when-do-we-go-for-pre-operat>

<http://stackoverflow.com/questions/23995802/crm-2013-plugins-can-some-one-explain-to-me-what-is-the-difference-in-message?rq=1>

The "message" is the actionable event that is taking place. On Create, a record is being inserted into CRM. For retrieve, a single record is being returned. Etc.

**Pipeline Stage** - Specifies when you want the plug-in to execute: before (Pre) or after (Post) the core operation that processes the message. For more information, see [Event execution pipeline](http://msdn.microsoft.com/en-us/library/gg327941.aspx).

**Pre Image Alias** - A pre-image is a snapshot of the entity’s attributes before the core operation.

**Post Image Alias** - A post-image is a snapshot of the entity’s attribute after the core operation.

If you are getting started, I'd suggest looking into the SDK. There are samples to get you started and help to explain a lot of this. <http://www.microsoft.com/en-us/download/details.aspx?id=24004>

|  |
| --- |
| [share](http://stackoverflow.com/a/23996607)[improve this answer](http://stackoverflow.com/posts/23996607/edit) |

Imp :

<http://inogic.com/blog/2010/07/pre-image-post-image-explained/>

Handle exceptions in plug-ins

# Handle exceptions in plug-ins

**Dynamics CRM 2015**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635832993609303621

* [Dynamics CRM 2013](https://msdn.microsoft.com/en-in/library/gg334685(v=crm.6).aspx)
* [Dynamics CRM 2011](https://msdn.microsoft.com/en-in/library/gg334685(v=crm.5).aspx)

Applies To: CRM 2015 on-prem, CRM Online

For synchronous plug-ins, whether registered in the sandbox or not, the Microsoft Dynamics CRM platform handles exceptions passed back from a plug-in by displaying an error message in a dialog of the web application user interface. The exception message for asynchronous registered plug-ins is written to a System Job (**AsyncOperation**) record which can be viewed in the System Jobs area of the web application.

For synchronous plug-ins, you can optionally display a custom error message in the error dialog of the web application by having your plug-in throw an [InvalidPluginExecutionException](https://msdn.microsoft.com/en-in/library/microsoft.xrm.sdk.invalidpluginexecutionexception.aspx) exception with the custom message string as the exception **Message** property value. If you throw [InvalidPluginExecutionException](https://msdn.microsoft.com/en-in/library/microsoft.xrm.sdk.invalidpluginexecutionexception.aspx) and do not provide a custom message, a generic default message is displayed in the error dialog. It is recommended that plug-ins only pass an [InvalidPluginExecutionException](https://msdn.microsoft.com/en-in/library/microsoft.xrm.sdk.invalidpluginexecutionexception.aspx) back to the platform.

If a synchronous plug-in returns an exception other than [InvalidPluginExecutionException](https://msdn.microsoft.com/en-in/library/microsoft.xrm.sdk.invalidpluginexecutionexception.aspx) back to the platform, the error dialog is displayed to the user and the exception message ([System.Exception.Message](https://msdn.microsoft.com/library/system.exception.message.aspx)) with stack trace is also written to one of two places. For plug-ins not registered in the sandbox, the information is written to the Application event log on the server that runs the plug-in. The event log can be viewed by using the Event Viewer administrative tool. For plug-ins registered in the sandbox, the exception message and stack trace is written to the Microsoft Dynamics CRM platform trace. For more information about tracing, see the Logging and Tracing section of the [Debug a plug-In](https://msdn.microsoft.com/en-in/library/gg328574.aspx) topic.

Error Logging Options in CRM

<http://salimadamoncrm.com/2013/08/14/error-logging-options-in-crm/>

<https://msdynamicscrmblog.wordpress.com/2012/07/01/log-error-details-from-the-plugins-in-dynamics-crm-2011/>

<http://www.magnetismsolutions.com/blog/ahmed-anwar's-blog/2015/06/04/tracing-and-logging-plugins-in-dynamics-crm-2015-online>

### Why impersonate System User

If running plugins as the calling user is so good, why impersonate other users or system admins.

* What if you need to retrieve records the user doesn’t have access to?
* What if you need to update records the user doesn’t have access to?

You might be thinking, “if the user doesn’t have access to those records, maybe the plugin shouldn’t be updating them”.

It’s a point to consider point but sometimes you want to create records or update records based on the action/status of an entity to move the code to the next stage/state.

Sometimes you want to assign a record to another user when a record goes to a certain state but you wouldn’t want users to be able to assign records.

## [Impersonate a user](javascript:void(0))

To impersonate a user, set the [CallerId](https://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationserviceproxy.callerid.aspx) property on an instance of [OrganizationServiceProxy](https://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationserviceproxy.aspx) before calling the service’s Web methods.

#### What is Impersonation

Impersonation in plugins/custom workflows is creating an IOrganisationService as a different user

#### Why use Impersonation

Use impersonation when you need to update/retrieve/delete/Create records users security roles doesn’t let them.

#### Where

If you don’t want to increase users security roles but still want the plugin to do a particularly action.  e.g. delete a record

#### When

You use impersonation in plugins/customer workflow

### Impersonation Code

Easy way to impersonate System User is to pass null to the CreateOrganizationService and it will run as System user, find out more in this [CRM SDK article](https://msdn.microsoft.com/en-us/library/gg309416.aspx)

// Use the factory to generate the Organization Service.  
OrganizationServiceImpersonated = factory.CreateOrganizationService(null);

<https://crmbusiness.wordpress.com/2015/07/21/crm-2015-understanding-impersonation-in-plugins-and-knowing-when-to-use-it/>