Platform SDK Developer's Guide

Creating an E-Mail

4/13/2025

Creating an E-Mail

Java

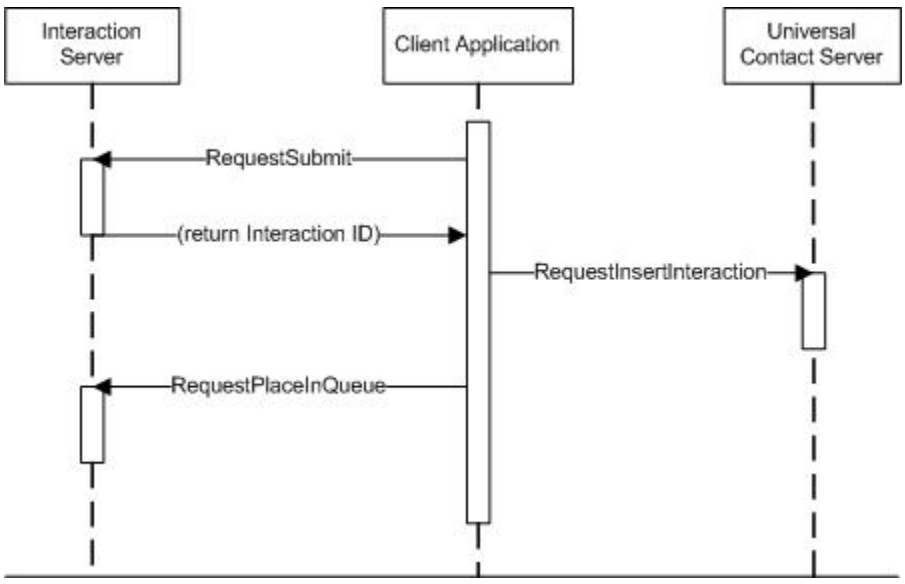
This article discusses the general process used to create e-mail messages, and provides suggestions about how you should work with those protocols.

Overview of Creating a New E-Mail Message

To create a new e-mail message, there are four basic steps you should follow:

1. Connect to Genesys Servers - Use the Protocol Manager Application Block to access the appropriate Genesys Servers.
2. Create a new Interaction - Request a new Interaction that will be used to manage the e-mail message within Interaction Server.
3. Store e-mail details in UCS - Once the Interaction is available, you can use the unique `InteractionId` that is returned to create a new UCS entry that contains details and contents for the e-mail message.
4. Place the Interaction in the appropriate queue - When both parts of the e-mail message have been stored, move the Interaction into the correct queue for processing.

A quick overview of these steps, and an outline of the key requests sent to Genesys servers, is shown below.



> **Tip**
> The order of the second and third steps can be reversed, if desired, as long as the final UCS entry contains the correct InteractionId value. In this case you would need to update the UCS entry after creating the new Interaction.

The following sections include code snippets that show one possible approach for handling each of these steps. The snippets have been simplified to focus only on code related to Genesys-specific functions.

Connecting to Genesys Servers

When creating and handling e-mail interactions, it is important to remember how e-mail messages are stored within the Genesys environment, and which Genesys servers you are interacting with.

Each e-mail message is stored as two separate pieces: an Interaction, and an entry in the Universal Contact Server (UCS) database. The e-mail is represented as an Interaction so that it can be sorted and processed using queues that have defined behavior. Even though e-mails are managed through Interaction Server, the actual contents and subject matter of each message must be stored in the UCS database. Any attempt to create or handle e-mail messages will require access to both Genesys Servers: Interaction Server (using the Open Media protocol) and UCS (using the Contacts Platform SDK protocol).

Before writing your e-mail application, some fairly standard code must be added to allow access to these Genesys servers. First, all necessary references and import statements must be added to your project. This includes the two specific protocols mentioned above, together with some common Genesys libraries and the Protocol Manager Application Block.

With those statements in place, we configure the Protocol Manager Application Block to handle communication with Genesys servers using the `ProtocolManagementServiceImpl` object, which is defined and configured as shown below.

```
[Java]

private InteractionServerProtocol interactionServerProtocol;
private UniversalContactServerProtocol contactServerProtocol;

public void connectToProtocols() throws URISyntaxException, ProtocolException
{
    Endpoint interactionServerEndpoint = new Endpoint(new URI("tcp://ixnServer:7005"));
    interactionServerProtocol = new InteractionServerProtocol(interactionServerEndpoint);
    interactionServerProtocol.setClientName("EmailSample");
    interactionServerProtocol.setClientType(InteractionClient.AgentApplication);

    Endpoint contactServerEndpoint = new Endpoint(new URI("tcp://ucsServer:7006"));
    contactServerProtocol = new UniversalContactServerProtocol(contactServerEndpoint);
    contactServerProtocol.setClientName("EmailSample");

    interactionServerProtocol.beginOpen();
    contactServerProtocol.beginOpen();
}
```

For more information about the Protocol Manager Application Block, see the Connecting to a Server article found in this guide.

Creating an Interaction

With connections to the Genesys servers established, we are ready to request a new Interaction that will represent our e-mail message in Interaction Server. You accomplish this by creating a new `RequestSubmit`, setting a few parameters to indicate that this Interaction represents an e-mail message, and then sending the request to Interaction Server with your `ProtocolManagementService` object.

```
[Java]

public void createInteraction(String ixnType, String ixnSubtype, String queue) throws
Exception
{
    RequestSubmit req = RequestSubmit.create();
    req.setInteractionType(ixnType);
    req.setInteractionSubtype(ixnSubtype);
    req.setQueue(queue);
    req.setMediaType("email");

    Message response = interactionServerProtocol.request(req);
    if(response == null || response.messageId() != EventAck.ID) {
          // For this sample, no error handling is implemented
          return;
    }

    EventAck event = (EventAck)response;
    mInteractionId = event.getExtension().getString("InteractionId");
}
```

A full list of properties that need to be set is included in the table below. Note that the `InteractionType` and `InteractionSubtype` properties must match existing business attributes, as specified in Configuration Server.

Property Name	Description
InteractionType	Interaction type for this e-mail message. Must match an Interaction Type Business Attribute, as specified in Configuration Server.
InteractionSubtype	Interaction subtype for this e-mail message. Must match an Interaction Subtype Business Attribute, as specified in Configuration Server.
Queue	Queue that this Interaction will be placed in initially. Must be defined in Configuration Server. When creating a new e-mail Interaction, the initial queue should not process the message (because additional information needs to be stored in UCS first).
MediaType	Primary media type of the interaction that is being submitted to Interaction Server. Intended for Media Server.

Once a response is received from Interaction Server, you can confirm that an `EventAck` response was returned and that the request was processed successfully. If an `EventError` response is returned instead, then you will need to implement some error handling code.

It is also important to save and track the `InteractionId` value of the newly created Interaction. This

ID needs to be specified in UCS entries that hold details related to the e-mail message, and is also required for moving the Interaction to an appropriate queue when you are ready to process the e-mail. In this example we are storing the `InteractionId` value in a simple variable named `mInteractionId`, which is assumed to be defined for your project. In larger samples (or full projects), a more robust way of tracking and handling Interactions may be required.

Storing E-Mail Details in UCS

With the ID of your newly created Interaction available, it is time to store details about the e-mail you are sending in the UCS database.

There are three types of information that must be stored in the UCS database:

- Interaction Attributes - Define details about the related Interaction for this information.
- Entity Attributes - Define where the e-mail message is coming from and going to. You will use `EmailOutEntityAttributes` for storing outbound e-mail messages, and `EmailInEntityAttributes` for storing inbound e-mail messages.
- Interaction Content - Define the actual contents of the email message, including the main text and any MIME attachments.

Creating and configuring a `RequestInsertInteraction` object with this information can be easily accomplished, as shown below.

```
[Java]

public void storeDetails(String ixnType, String ixnSubtype) throws Exception
{
    // Set Interaction Attributes
    InteractionAttributes ixnAttributes = new InteractionAttributes();
    ixnAttributes.setId(mInteractionId);
    ixnAttributes.setMediaTypeId("email");
    ixnAttributes.setTypeId(ixnType);
    ixnAttributes.setSubtypeId(ixnSubtype);
    ixnAttributes.setTenantId(101);
    ixnAttributes.setStatus(Statuses.Pending);
    ixnAttributes.setSubject("Sample e-mail subject");
    ixnAttributes.setEntityTypeId(EntityTypes.EmailOut);

    // Set Entity Attributes
    EmailOutEntityAttributes entityAttributes = new EmailOutEntityAttributes();
    entityAttributes.setFromAddress("sending@email.com");
    entityAttributes.setToAddresses("receiving@email.com");
    entityAttributes.setCcAddresses("copying@email.com");
    ...

    // Set Interaction Content
    InteractionContent content = new InteractionContent();
    content.setText("This is the e-mail body.");
    ...

    // Send the request
    RequestInsertInteraction req = new RequestInsertInteraction();
    req.setInteractionAttributes(ixnAttributes);
    req.setEntityAttributes(entityAttributes);
    req.setInteractionContent(content);

    contactServerProtocol.send(req);
```

```
}
```

A list of `InteractionAttributes` properties that need to be set for an email message is provided in the following table. The properties shown for `EmailOutEntityAttributes` and `InteractionContent` represent some of those most commonly used with email. Please check the documentation provided for each class to see a full list of available properties.

Interaction Attribute Name	Description
EntityTypeId	Indicates whether this is an outgoing or incoming e-mail.
Id	Interaction ID of the related Interaction record, created earlier.
MediaTypeId	Primary media type of the Interaction you are submitting to Interaction Server. Intended for Media Server.
Subject	Subject line for this e-mail message.
SubtypeId	Interaction subtype for this e-mail message. Must match an Interaction Subtype Business Attribute, as specified in Configuration Server.
Status	Current status of the e-mail message.
TenantId	ID of the Tenant where this e-mail belongs.
TypeId	Interaction type for this e-mail message. Must match an Interaction Type Business Attribute, as specified in Configuration Server.

Placing the Interaction in the Appropriate Queue

When an Interaction has been created to handle the e-mail, and all content has been stored in the UCS database, you are free to begin processing the message as you would process any normal Interaction. This is accomplished by moving the Interaction that you created into the appropriate queue for e-mail processing, as defined in Interaction Routing Designer.

```
[Java]

public void placeInQueue(String queue) throws Exception
{
    RequestPlaceInQueue req = RequestPlaceInQueue.create();
    req.setInteractionId(mInteractionId);
    req.setQueue(queue);

    interactionServerProtocol.send(req);
}
```

Replying to an E-Mail Message

Replying to an existing e-mail message follows the same basic process outlined above, but requires a few additional parameters to be set in your requests. These changes are described in the following subsections.

Changes to Creating an Interaction

When creating the Interaction, you need to specify one additional parameter before submitting your `RequestSubmit`. Take the `InteractionId` of the Interaction that represents the original e-mail message, and use that value as the `ParentInteractionId` parameter in your request, as shown below:

```
[Java]

RequestSubmit req = RequestSubmit.create();

...

req.setParentInteractionId = parentInteractionId;
```

The following table describes these additional attributes.

Attribute Name	Description
ParentInteractionId	`InteractionId` of a parent e-mail Interaction. Only set this value when replying to an existing e-mail message.

Changes to Storing E-Mail Details in UCS

When storing e-mail details in UCS, you need to specify values for three additional interaction attributes before sending your `RequestInsertInteraction`. These attributes (shown in the code snippet below) provide a link between the parent entry in UCS and any related children, as well as specifying a common thread ID.

```
[Java]

InteractionAttributes ixnAttributes = new InteractionAttributes();

...

ixnAttributes.setParentId(parentInteractionId);
ixnAttributes.setCanBeParent(False);
ixnAttributes.setThreadId(parentThreadId);
```

The table below describes these additional attributes.

Attribute Name	Description
CanBeParent	Boolean value that indicates whether this message can be a parent.
ParentId	Interaction ID for the parent e-mail Interaction.
ThreadId	Unique value that is shared between all UCS entries in an e-mail conversation.

Other Considerations

Although this introduction to creating and handling e-mail messages is not intended to be a comprehensive guide, it is useful to quickly introduce some other considerations and basic concepts regarding how requests are submitted and how errors should be handled.

- The first consideration to take into account is how you submit requests using the Protocol Management Application Block. In the code provided here, a simple send method is used to submit most requests without waiting for a response from the server. However, in more complicated samples or implementations you may need to process responses, or store and use values returned (such as the `InteractionId` in this example) once a request is processed.
  Please read the article on Event Handling provided in this document for a better understanding of how to handle incoming responses in both a synchronous and asynchronous fashion. This allows better error handling to be implemented if a request fails.
- A second consideration to be aware of is how records in Interaction Server and UCS are related when implementing error handling. If you have already created a new Interaction when your `RequestInsertInteraction` fails, then you will need to either resubmit the UCS record or delete the related Interaction by submitting a `RequestStopProcessing`. (If you reversed the steps shown here and created a UCS record first, then the same concept applies for removing that record when a new Interaction request fails.)

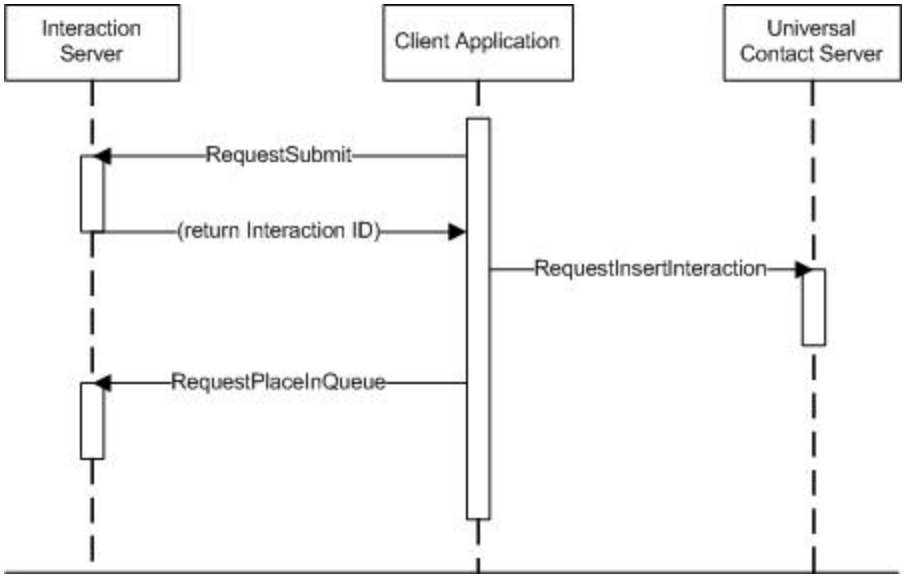
.NET

Overview of Creating a New E-Mail Message

To create a new e-mail message, there are four basic steps you should follow:

1. Connect to Genesys Servers - Use the Protocol Manager Application Block to access the appropriate Genesys Servers.
2. Create a new Interaction - Request a new Interaction that will be used to manage the e-mail message within Interaction Server.
3. Store e-mail details in UCS - Once the Interaction is available, you can use the unique InteractionId that is returned to create a new UCS entry that contains details and contents for the e-mail message.
4. Place the Interaction in the appropriate queue - When both parts of the e-mail message have been stored, move the Interaction into the correct queue for processing.

A quick overview of these steps, and an outline of the key requests sent to Genesys servers, is shown below.



> **Tip**
>
> The order of the second and third steps can be reversed, if desired, as long as the final UCS entry contains the correct `InteractionId` value. In this case you would need to update the UCS entry after creating the new Interaction.

The following sections include code snippets that show one possible approach for handling each of these steps. The snippets have been simplified to focus only on code related to Genesys-specific functions.

Connecting to Genesys Servers

When creating and handling e-mail interactions, it is important to remember how e-mail messages are stored within the Genesys environment, and which Genesys servers you are interacting with.

Each e-mail message is stored as two separate pieces: an Interaction, and an entry in the Universal Contact Server (UCS) database. The e-mail is represented as an Interaction so that it can be sorted and processed using queues that have defined behavior. Even though e-mails are managed through Interaction Server, the actual contents and subject matter of each message must be stored in the UCS database. Any attempt to create or handle e-mail messages will require access to both Genesys Servers: Interaction Server (using the Open Media protocol) and UCS (using the Contacts Platform SDK protocol).

Before writing your e-mail application, some fairly standard code must be added to allow access to these Genesys servers. First, all necessary references and using statements must be added to your project.

```
[C#]

private InteractionServerProtocol interactionServerProtocol;
private UniversalContactServerProtocol contactServerProtocol;
```

```
public void ConnectToProtocols()
{
        var interactionServerEndpoint = new Endpoint(new Uri("tcp://ixnServer:7005"));
        interactionServerProtocol = new InteractionServerProtocol(interactionServerEndpoint);
        interactionServerProtocol.ClientName = "EmailSample";
        interactionServerProtocol.ClientType = InteractionClient.AgentApplication;

        var contactServerEndpoint = new Endpoint(new Uri("tcp://ucsServer:7006"));
        contactServerProtocol = new UniversalContactServerProtocol(contactServerEndpoint);
        contactServerProtocol.ClientName = "EmailSample";

        interactionServerProtocol.BeginOpen();
        contactServerProtocol.BeginOpen();
}
```

Creating an Interaction

With connections to the Genesys servers established, we are ready to request a new Interaction that will represent our e-mail message in Interaction Server. All you need to do to accomplish this is to create a new `RequestSubmit`, set a few parameters to indicate that this Interaction represents an e-mail message, and then use your `InteractionServerProtocol` object to send that request to Interaction Server.

Unlike other requests shown in this article, `RequestSubmit` is sent using the `BeginRequest` method so that we can receive and process the response from Interaction Server.

```
[C#]

public void CreateInteraction(string ixnType, string ixnSubtype, string queue)
{
var req = RequestSubmit.Create();
req.InteractionType = ixnType;
req.InteractionSubtype = ixnSubtype;
req.MediaType = "email";
req.Queue = queue;

interactionServerProtocol.BeginRequest(req, new AsyncCallback(OnCreateInteractionComplete),
null);
}
```

A full list of properties that need to be set is included in the following table. Note that the `InteractionType` and `InteractionSubtype` properties must match existing business attributes, as specified in Configuration Server.

Property Name	Description
InteractionSubtype	Interaction subtype for this e-mail message. Must match an Interaction Subtype Business Attribute, as specified in Configuration Server.
InteractionType	Interaction type for this e-mail message. Must match an Interaction Type Business Attribute, as specified in Configuration Server.
MediaType	Primary media type of the interaction that is being submitted to Interaction Server. Intended for Media

Property Name	Description
	Server.
Queue	Queue that this Interaction will be placed in initially. Must be defined in Configuration Server. When creating a new e-mail Interaction, the initial queue should not process the message (because additional information needs to be stored in UCS first).

Once a response is received from Interaction Server, you can confirm that an `EventAck` response was returned and that the request was processed successfully. If an `EventError` response is returned instead, then you will need to implement some error handling code.

You should also save and track the `InteractionId` value of the newly created Interaction. This ID needs to be specified in UCS entries that hold details related to the e-mail message, and is also required for moving the Interaction to an appropriate queue when you are ready to process the e-mail.

```
[C#]

private void OnCreateInteractionComplete(IAsyncResult result)
{
    var response = interactionServerProtocol.EndRequest(result);
    if (response == null || response.Id != EventAck.MessageId)
        // for this sample, no error handling is implemented
        return;

    var @event = response as EventAck;
    mInteractionId = (string)@event.Extension["InteractionId"];
}
```

In this example we are storing the `InteractionId` value in a simple variable named `mInteractionId`, which is assumed to be defined for your project. In larger samples (or full projects), a more robust way of tracking and handling Interactions may be required.

Storing E-Mail Details in UCS

With the ID of your newly created Interaction available, it is time to store details about the e-mail you are sending in the UCS database.

There are three types of information that must be stored in the UCS database:

- Interaction Attributes - Define details about the related Interaction for this information.
- Entity Attributes - Define where the e-mail message is coming from and going to. You will use EmailOutEntityAttributes for storing outbound e-mail messages, and EmailInEntityAttributes for storing inbound e-mail messages.
- Interaction Content - Define the actual contents of the email message, including the main text and any MIME attachments.

Creating and configuring a `RequestInsertInteraction` object with this information can be easily accomplished, as shown below.

```
[C#]
```

```
public void StoreDetails(string ixnType, string ixnSubtype)
{
    var req = new RequestInsertInteraction();
    req.InteractionAttributes = new InteractionAttributes()
    {
        Id = mInteractionId,
        MediaTypeId = "email",
        TypeId = ixnType,
        SubtypeId = ixnSubtype,
        TenantId = 101,
        Status = new NullableStatuses(Statuses.Pending),
        Subject = "Sample e-mail subject",
        EntityTypeId = new NullableEntityTypes(EntityTypes.EmailOut),
    };
    req.EntityAttributes = new EmailOutEntityAttributes()
    {
        FromAddress = "sending@email.com",
        ToAddresses = "receiving@email.com",
        CcAddresses = "copied@email.com",
        ...
    };
    req.InteractionContent = new InteractionContent()
    {
        Text = "This is the e-mail body.",
       ...
    };
    contactServerProtocol.Send(req);
}
```

A list of `InteractionAttributes` properties that need to be set for an email message is provided in the following table. The properties shown for `EmailOutEntityAttributes` and `InteractionContent` represent some of those most commonly used with email. Please check the documentation provided for each class to see a full list of available properties.

Interaction Attribute Name	Description
EntityTypeId	Indicates whether this is an outgoing or incoming e-mail.
Id	Interaction ID of the related Interaction record, created earlier.
MediaTypeId	Primary media type of the Interaction you are submitting to Interaction Server. Intended for Media Server.
Subject	Subject line for this e-mail message.
SubtypeId	Interaction subtype for this e-mail message. Must match an Interaction Subtype Business Attribute, as specified in Configuration Server.
Status	Current status of the e-mail message.
TenantId	ID of the Tenant where this e-mail belongs.
TypeId	Interaction type for this e-mail message. Must match an Interaction Type Business Attribute, as specified in Configuration Server.

Placing the Interaction in the Appropriate Queue

When an Interaction has been created to handle the e-mail, and all content has been stored in the UCS database, you are free to begin processing the message as you would process any normal Interaction. This is accomplished by moving the Interaction that you created into the appropriate queue for e-mail processing, as defined in Interaction Routing Designer.

```
[C#]

public void PlaceInQueue(string queue)
{
    var req = RequestPlaceInQueue.Create();
    req.InteractionId = mInteractionId;
    req.Queue = queue;

    interactionServerProtocol.Send(req);
}
```

Replying to an E-Mail Message

Replying to an existing e-mail message follows the same basic process outlined above, but requires a few additional parameters to be set in your requests. These changes are described in the following subsections.

Changes to Creating an Interaction

When creating the Interaction, you need to specify one additional parameter before submitting your `RequestSubmit`. Take the `InteractionId` of the Interaction that represents the original e-mail message, and use that value as the `ParentInteractionId` parameter in your request, as shown below:

```
[C#]

var req = RequestSubmit.Create();
...
req.ParentInteractionId = parentInteractionId;
```

The following table describes these additional attributes.

Attribute Name	Description
ParentInteractionId	InteractionId of a parent e-mail Interaction. Only set this value when replying to an existing e-mail message.

Changes to Storing E-Mail Details in UCS

When storing e-mail details in UCS, you need to specify values for three additional interaction attributes before sending your `RequestInsertInteraction`. These attributes (shown in the code snippet below) provide a link between the parent entry in UCS and any related children, as well as specifying a common thread ID.

```
[C#]

var req = new RequestInsertInteraction();
...
req.InteractionAttributes.ParentId = parentInteractionId;
req.InteractionAttributes.CanBeParent = False;
req.InteractionAttributes.ThreadId = parentThreadId;
```

The following table describes these additional attributes.

Attribute Name	Description
CanBeParent	Boolean value that indicates whether this message can be a parent.
ParentId	Interaction ID for the parent e-mail Interaction.
ThreadId	Unique value that is shared between all UCS entries in an e-mail conversation.

Other Considerations

Although this introduction to creating and handling e-mail messages is not intended to be a comprehensive guide, it is useful to quickly introduce some other considerations and basic concepts regarding how requests are submitted and how errors should be handled.

The first consideration to take into account is how you submit requests. In the code provided here, a simple Send method is used to submit most requests without waiting for a response from the server. However, for more complicated samples or implementations you should consider using the `BeginRequest` method with a callback handler instead.

Using `BeginRequest` allows requests to be submitted without waiting for a response, but provides the ability to confirm the result and response of each request. This allows better error handling to be implemented if a request fails. *Creating an Interaction* uses the `BeginRequest` method and a callback handler to capture the InteractionID that is returned.

A second consideration to be aware of is how records in Interaction Server and UCS are related when implementing error handling. If you have already created a new Interaction and then the `RequestInsertInteraction` fails, you need to either resubmit the UCS record or delete the related Interaction by submitting a `RequestStopProcessing`. (If you reversed those steps and created a UCS record first, then the same idea must be applied if the request to create a new Interaction fails.)