Capturing Data Changes with a Harmony API or HTTP Endpoint

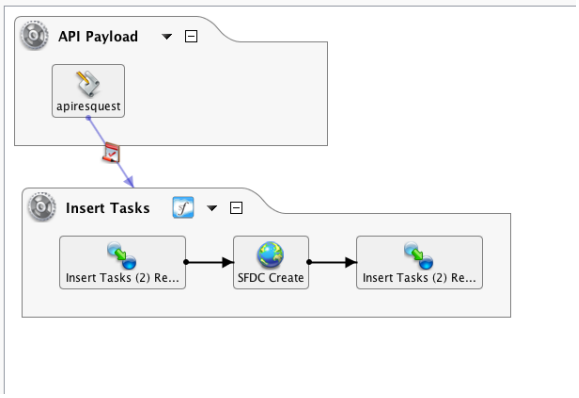
Use Case

In this pattern, instead of Jitterbit querying or reading a source, an external source is sending a payload to Jitterbit, where an operation acts as a host. Typically the payload is a single record that is processed immediately through to the target, which is different from a periodic query dealing with a batch of records. Consideration has to be given as to scalability, in terms of the number of records that can be received within a short period of time, as well as using asynchronous calls if an acknowledgement back to the source is required.

> **NOTE:** This design pattern uses Design Studio as an example; you may apply the same concepts in Cloud Studio using similar steps.

Example 1 - API

Here is an example where the entire payload is sent instead of just an ID. In this case, the source, Five9, is sending a payload to the Jitterbit API Platform. The first operation receives the data, the second inserts the data into SFDC.



The internal Jitterbit variable is used to get the fields (named i, a, d, etc.) and assign them to global variables, then run the SFDC insert operation:

```
$i=$jitterbit.api.request.body.i;
$a=$jitterbit.api.request.body.a;
$d=$jitterbit.api.request.body.d;
$e=$jitterbit.api.request.body.e;
$f=$jitterbit.api.request.body.f;
$h=$jitterbit.api.request.body.h;
$k=$jitterbit.api.request.body.k;
$l=$jitterbit.api.request.body.l;
$m=$jitterbit.api.request.body.m;
$n=$jitterbit.api.request.body.n;
$p=$jitterbit.api.request.body.p;
$q=$jitterbit.api.request.body.q;
$r=$jitterbit.api.request.body.r;
$s=$jitterbit.api.request.body.s;
$t=$jitterbit.api.request.body.t;
$u=$jitterbit.api.request.body.u;
$y=$jitterbit.api.request.body.y;
RunOperation("<TAG>Operations/Inbound from Five9/Insert Tasks</TAG>",false)
```

The global variables are assigned to the transformation and processed.

On This Page

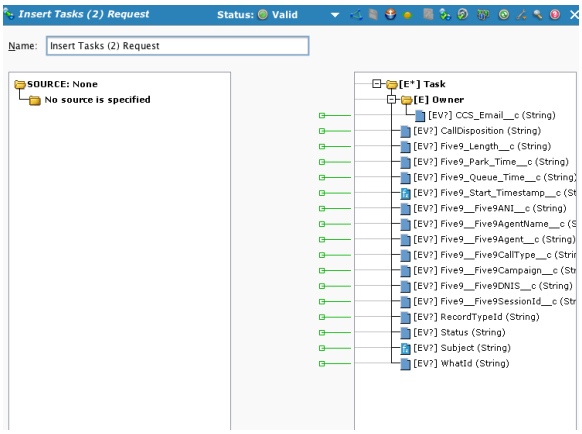
Related Articles

- Activities
- API Jitterbit Variables
- Configuring Outbound Messages with Harmony API
- Configuring Outbound Messages with Hosted HTTP Endpoints
- Jitterbit Log File Locations

Related Topics

- API Manager
- Design Pattern Library
- Design Studio
- Formula Builder
- Scripting

Last updated: Oct 04, 2019

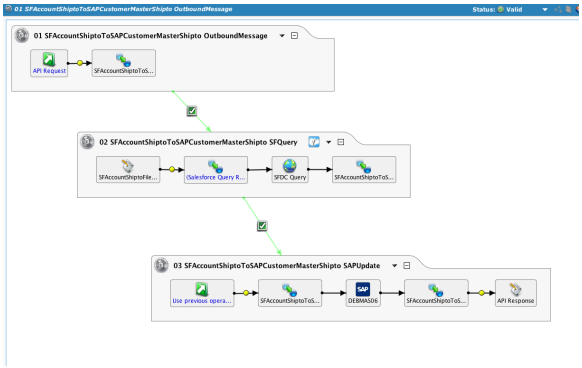


This pattern is less complex than the patterns in Capturing Data Changes with Timestamp-based Queries or Capturing Data Changes with Source Field Values. Obviously, the business rules related to when to send data in this case are embedded in the source application. Otherwise, the integration tool has to incorporate them, which leads to more complexity. Regardless of the approach, Jitterbit can accommodate it.

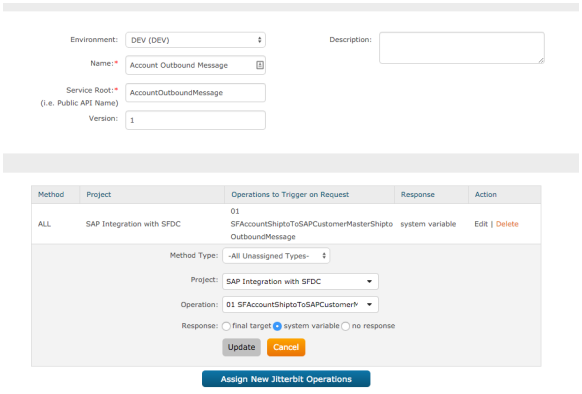
Example 2 - API

Here is another example using the Jitterbit API Platform.

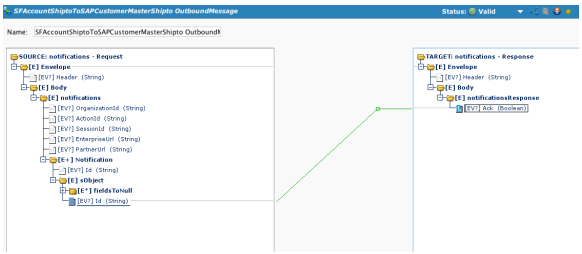
> **TIP:** For an additional example with step-by-step instructions, see Configuring Outbound Messages with Harmony API.



An outbound message is received from SFDC containing a single ID, which is passed to a query, and in turn, an SAP IDoc is sent to SAP.



Taking a look at the API configuration, the project name and the specific operation is selected. Also, note that 'system variable' is checked, which is the response back to the outbound message.



The transformation captures the inbound IDs. An outbound message can contain more than one record.

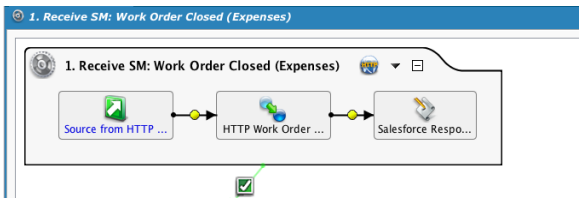
```
<trans>

IdCount = Count(Envelope$Body$notifications$Notification#.sObject$Id$);
i = 0;
While( i < IdCount,
    accountId=FindByPos((i+1), Envelope$Body$notifications$Notification#.sObject$Id$);
    WriteToOperationLog($accountId);
    RunOperation("<TAG>Operations/SFProspectToSAPShipTo/02 SFAccountShiptoToSAPCustomerMasterShipto SFQuery</TAG>");
    i++;
);

resp='<?xml version="1.0" encoding="UTF-8"?><Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://soap.sforce.com/2005/09/outbound">
<Body>
<ns:notificationsResponse>
<ns:Ack>1</ns:Ack>
</ns:notificationsResponse>
</Body>
</Envelope>
';
$jitterbit.api.response= resp
</trans>
```

The logic in this script will cycle through each ID, assign it to the global variable $accountId, and pass it to the next operation. The system variable (as called for in the API configuration) is assigned to the standard XML expected by the outbound message.

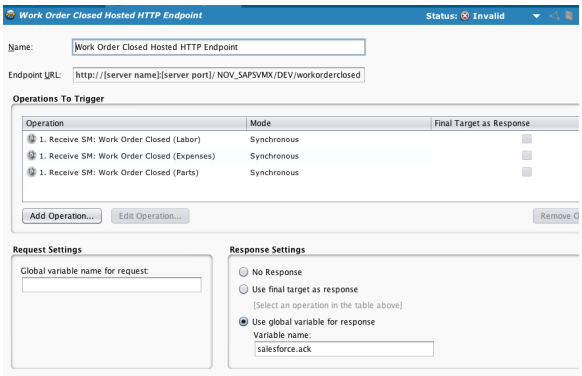
Example 3 - HTTP

The source system is SFDC and the target is SAP. The Jitterbit operation is using an HTTP Endpoint. An alternative to using an HTTP Endpoint with a Private Agent (and avoid opening ports in your firewalls) is to use the Jitterbit API platform, which is designed to handle high-traffic scenarios and is the much preferred method.

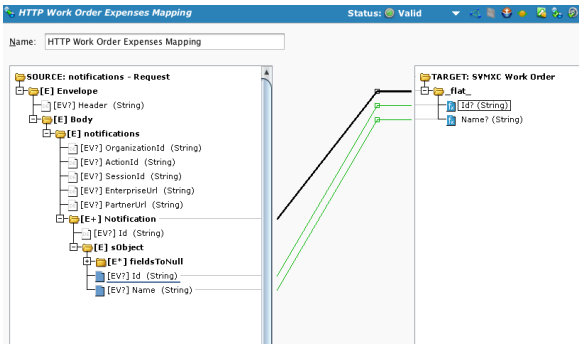
This operation is configured to use an HTTP Endpoint and respond with an acknowledgement.



The HTTP endpoint is configured to respond with a global variable. On a side note, if SFDC does not receive a response, the outbound message will be resent after a period of time. It is a good practice to send the acknowledgement back as soon as possible.



The transformation maps the inbound XML to a flat file structure.



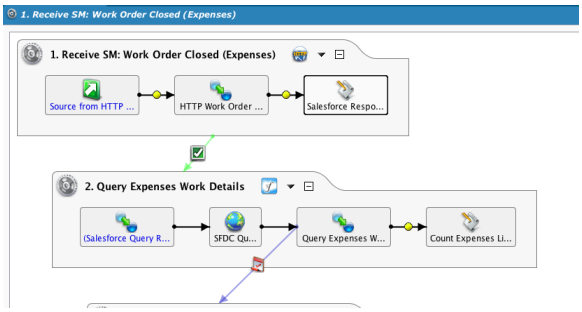
In the "Id" node, global variables are assigned, and information is written to the Operation Log.

```
$WorkOrderId = Envelope$Body$notifications$Notification.sObject$Id$;
$QuotedWorkOrderId = Quote($WorkOrderId);
$WorkOrderIdsQueryString = $WorkOrderIdsQueryString + $QuotedWorkOrderId +
",";
$WorkOrderIdsClean = RTrimChars($WorkOrderIdsQueryString, ",");
WriteToOperationLog(" ");
WriteToOperationLog("(" + TargetInstanceCount() + ")");
WriteToOperationLog("Work Order Id: " + $WorkOrderId);
WriteToOperationLog("Ids String: " + $WorkOrderIdsQueryString);
WriteToOperationLog("Clean Ids String: " + $WorkOrderIdsClean);
$WorkOrderId;
```

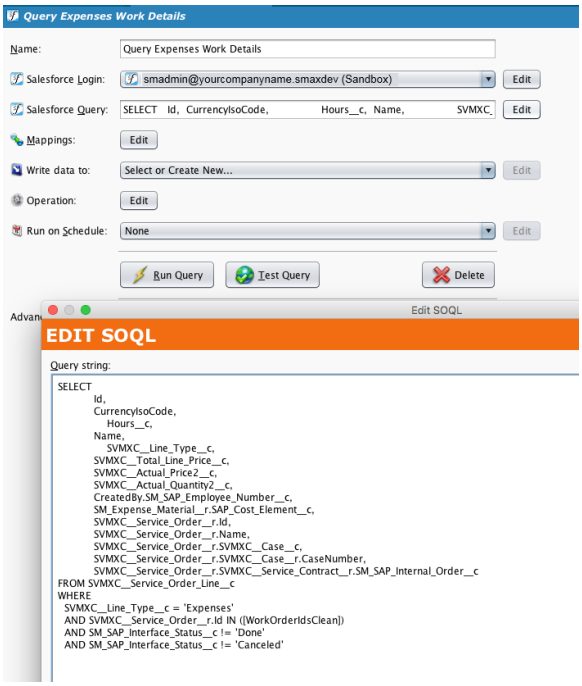
The post-operation script loads the acknowledgement into a global variable, which the HTTP endpoint is configured to pass back.

```
$salesforce.ack=
'<?xml version="1.0" encoding="UTF-8"?><Envelope xmlns="http://schemas.
xmlsoap.org/soap/envelope/" xmlns:ns="http://soap.sforce.com/2005/09
/outbound">
 <Body>
 <ns:notificationsResponse>
 <ns:Ack>1</ns:Ack>
 </ns:notificationsResponse>
 </Body>
 </Envelope>
 ';
```

After the first operation runs, a Salesforce query runs using the Id from the payload.



Here, the global variable from the first operation is used with the '[]' notation:



This is a common pattern. Instead of loading the incoming payload with all the data, only an ID to the record is passed and used in a query. It is also common to use global variables instead of writing temp files, which is more common with periodic queries and batches of data. Since global variables are specific to chained operations, even if messages are being received faster than a single chain can complete, the variable values will not be overwritten. If temp files were used, that may be an issue unless extra measures were taken, such as appending a guid to the filename, to make the files unique.