

Legend for Diagrams

Hook Flow diagrams



Black arrows indicate normal flow.



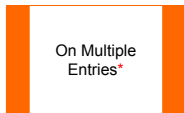
Red arrows show error/exception flow. Errors can occur both in scripted flow components, as well as in Integrator operations.



The Flow Endpoint symbol represents the start or end of the flow for a Connector flow diagram. The text contained in the symbol provides more information about system state and behavior at this point.



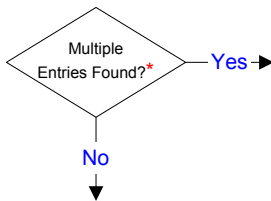
These boxes represent scripted flow components, and are used for both Attribute Maps and Hooks. Note that if a Hook is enabled, then control is passed to the script in the Hook. If a Hook is not enabled, then the flow continues past the Hook without executing it.



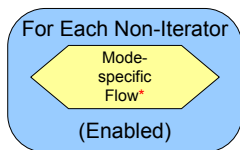
A few Hooks are *mandatory* and must be enabled, although they do not need to actually contain any script. If a mandatory Hook is not enabled and the flow reaches this point, then this is considered an **error**, and control faults out to error handling.



This box represents an Integrator operation (these are available as functions in the Raw Connector object. Note that Integrator operations may also result in error flows.



Decision components represent logical branches in Connector execution, depending on state information at this point in the flow.



The yellow trapezoid describe flow which is detailed elsewhere (i.e. on another page in this document. The optional rounded blue box includes instructions (as in the case show here where the specified flow is for each Connector in the AssemblyLine that is not in Iterator mode.



The Continuation symbol indicates that the flow is continued on another page that is common for one or more modes. The page being referenced will appear in a label below this symbol.



This is a Continuation symbol that is used when the referenced page is still part of the same Connector mode. The page being referenced will appear in a label below this symbol.

AssemblyLine Flow

Hook Flow diagrams

*Flow References

These yellow trapezoids represent flows found in the AssemblyLine Connectors.

Initialization Flows are found on the pages entitled **Initialization & Close Flows**

Iterator Flow is described on the page for **Iterator Mode** flow.

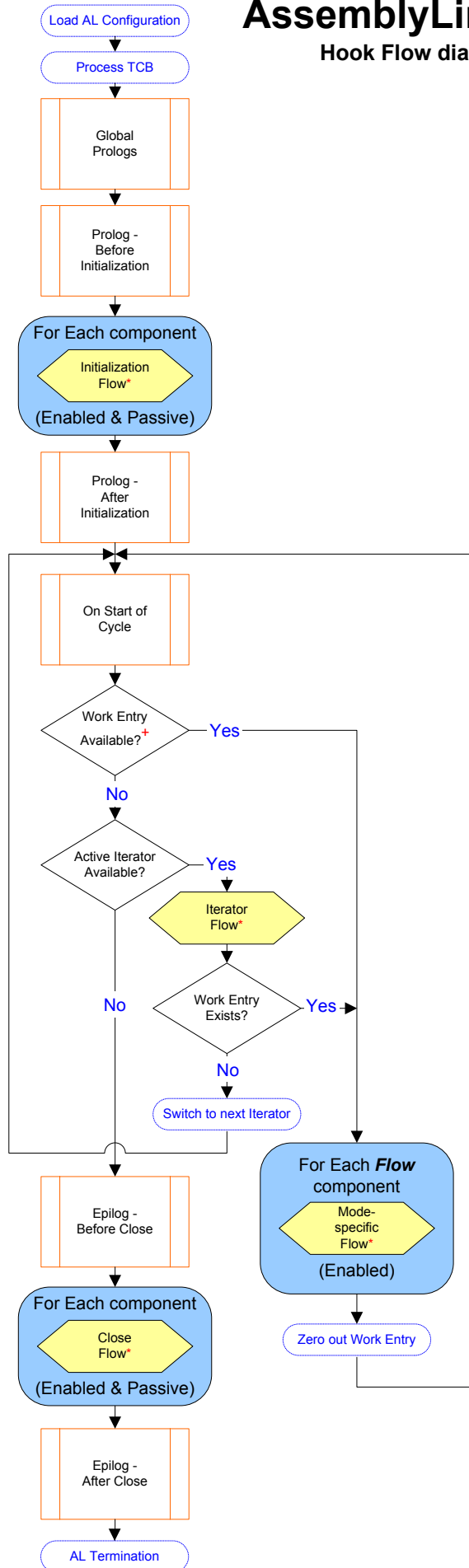
Mode-specific Flow can be found on the page(s) for that Connector Mode.

+Work Entry Available

This test checks to see if there is an **Entry** object which is to be used as **work** for the new cycle.

This Entry can be provided in a number of ways:

- o an **Initial Work Entry (IWE)**
- o via a call to `task.setWork()`
- o using `system.restartEntry()`



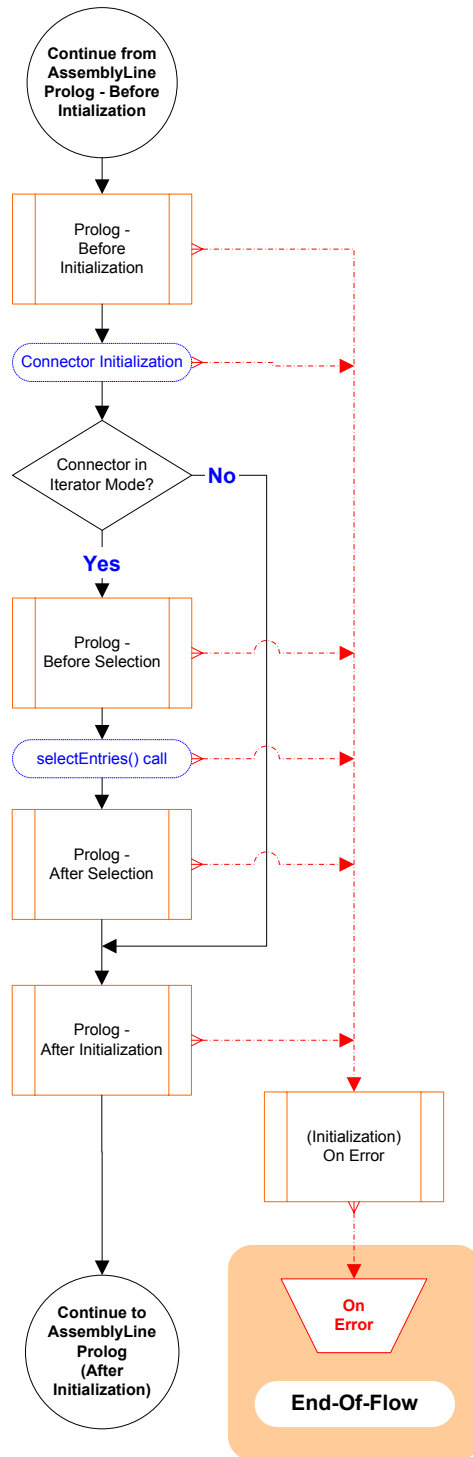
Connector Initialization Flow

Hook Flow diagrams

Available Objects

The **work** object is not available in Initialization Hooks (unless it is provided as an **Initial Work Entry (IWE)**).

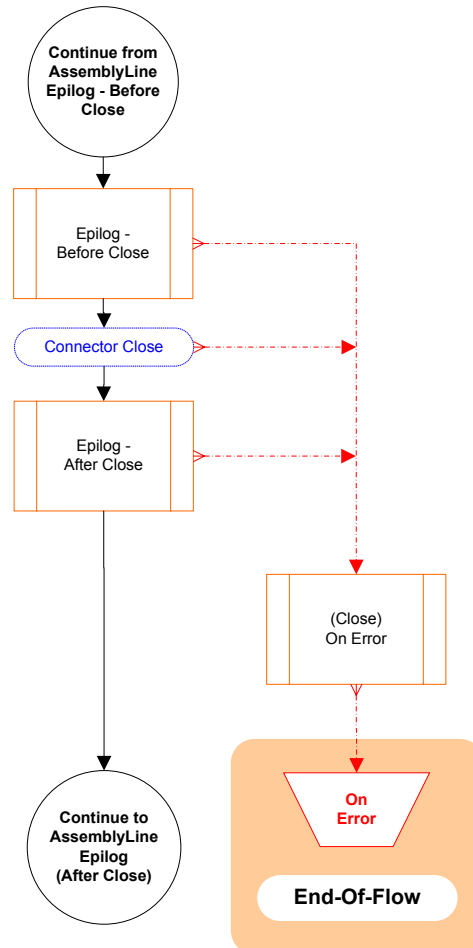
As always, if an **Error Hook** is enabled, the error flow continues and does not go to the **Error Flow**.



Available Objects

Close Hooks will have access to the last **work** Entry processed by the AssemblyLine

As always, if an **Error Hook** is enabled, the error flow continues and does not go to the **Error Flow**.



Available Objects

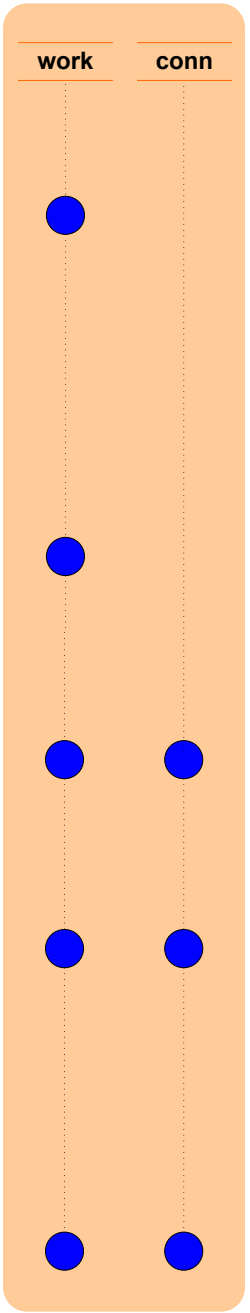
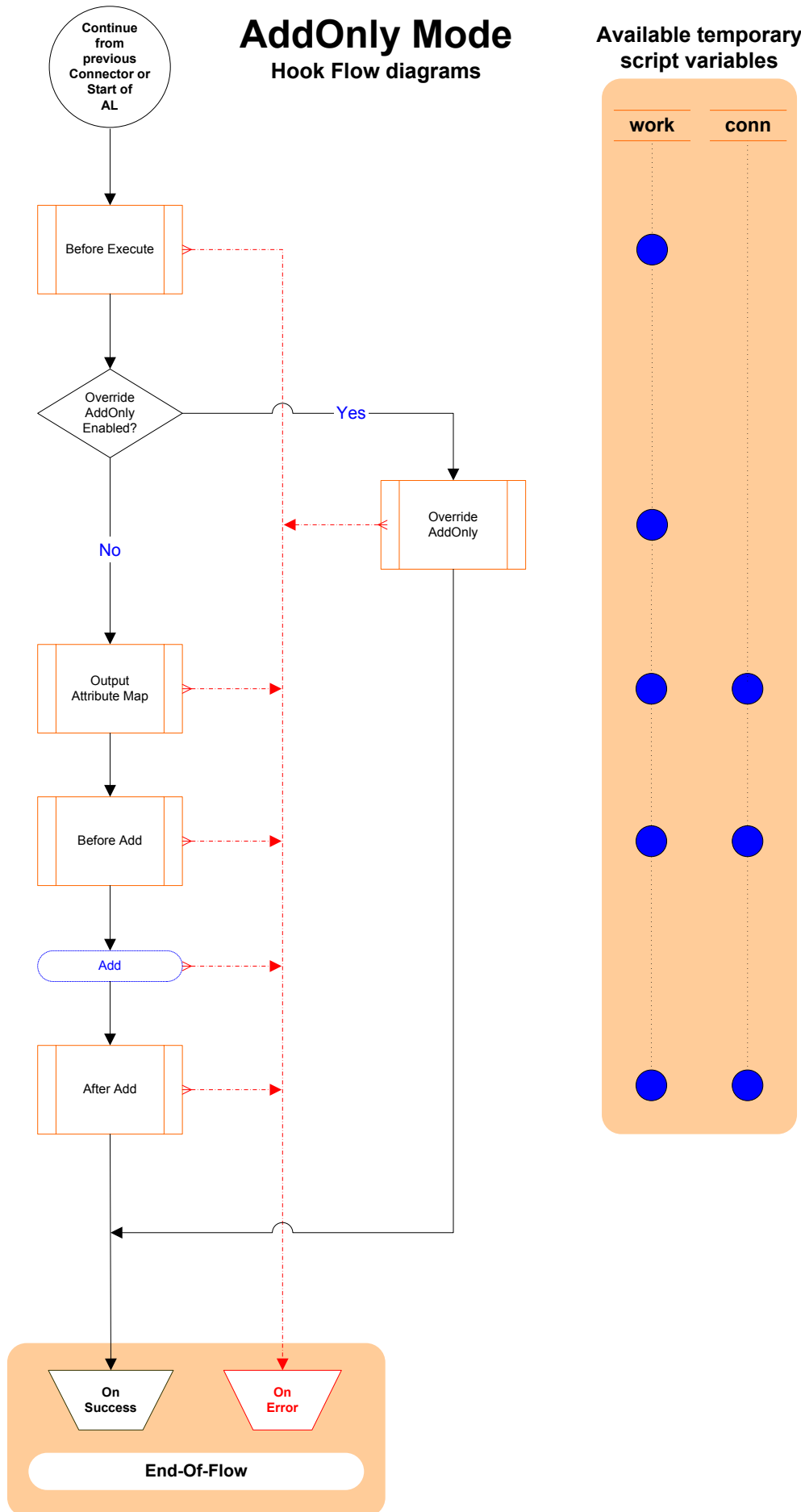
As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

The information stored in the **conn** object is written to the data source by the **Add** operation.

AddOnly Mode

Hook Flow diagrams

Available temporary script variables



Call/Reply Mode

Hook Flow diagrams

Available temporary script variables

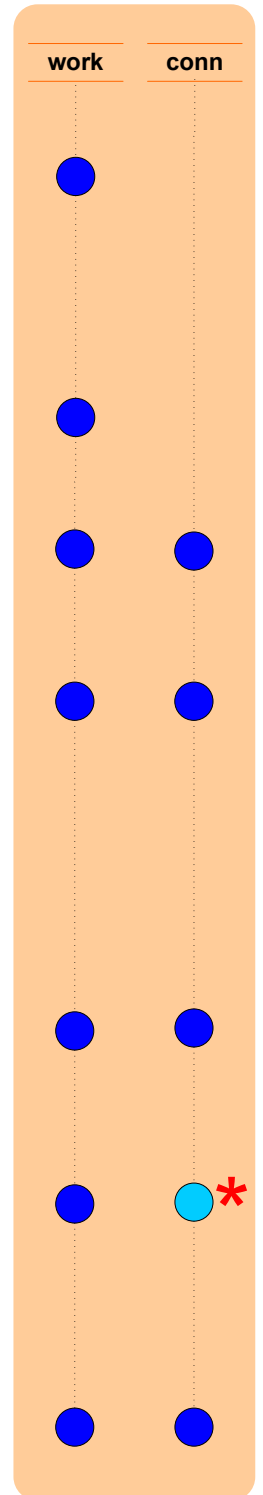
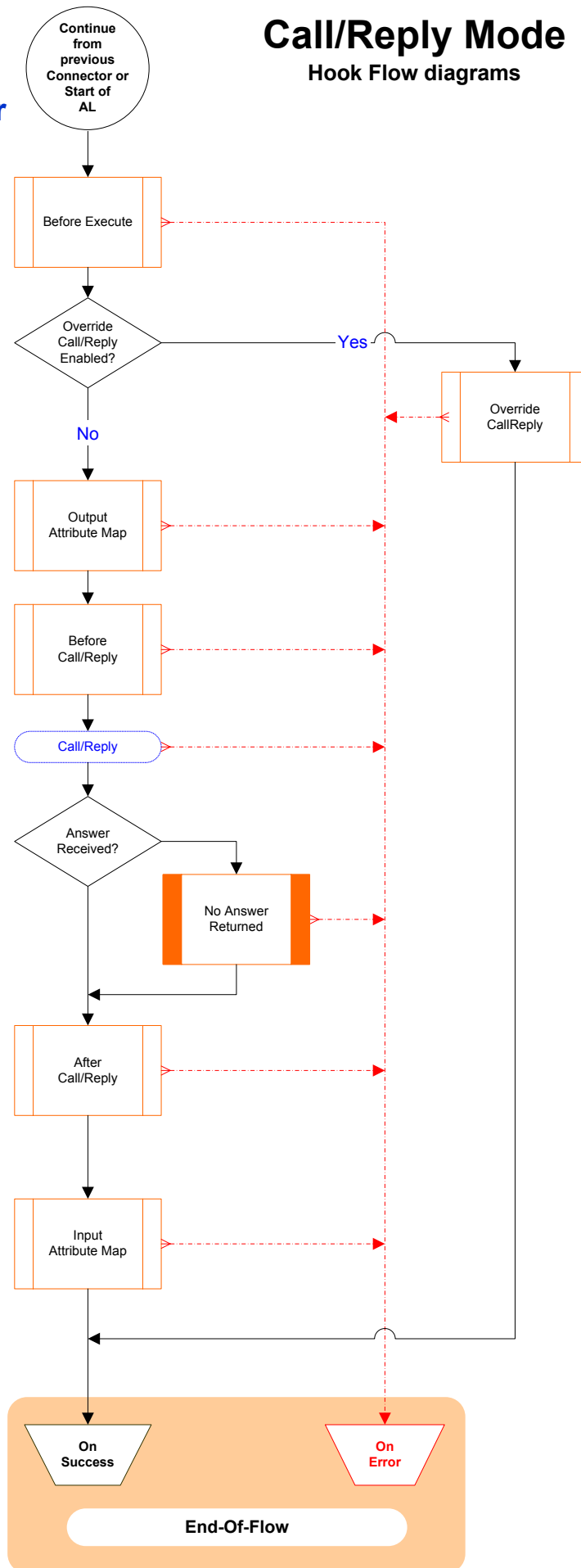
Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

*The information stored in the **conn** object is slightly different in this mode.

It is important to note that the **conn** object serves two different purposes in **Call/Reply** mode:

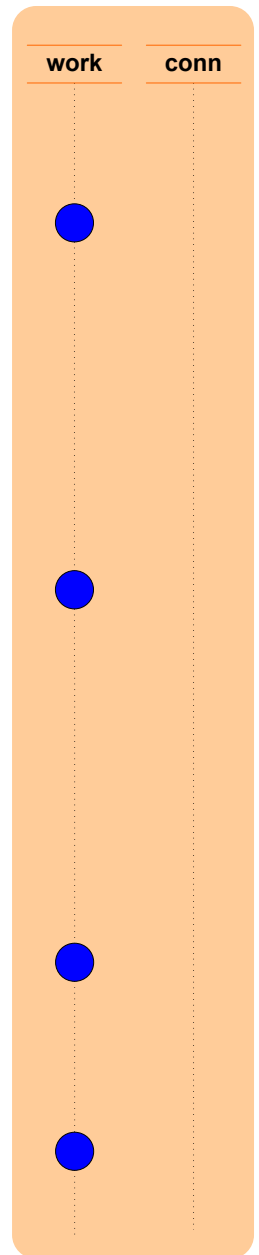
- 1) Storing the call attributes/parameters defined in the **Output Attribute Map** to be transmitted by the Call/Reply operation,
- 2) Receiving return attributes/parameters that will be accessed by the **Input Attribute Map** after the Call/Reply operation



Delete Mode 1/2

Hook Flow diagrams

Available temporary script variables



Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

After the **Build Link Criteria** operation, there is a script object called **search** available which gives you access to this information (i.e. for use in the Override Hook).

The record/entry matching the **Link Criteria** (and that is about to be deleted) is available for scripting as the **conn** object, and Attribute Mapping is carried out to allow your AssemblyLine to use Attributes from the Entry which is to be deleted.

***On Multiple Entries**

If more than one record/entry is found that matches the Link Criteria then the On Multiple Entries Hook must also be **enabled**, or this is treated as an **error**.

You can access the set of records/entries found by using either of these two Connector functions:

```
getFirstDuplicateEntry()
or
getNextDuplicateEntry()
```

Each of these functions returns an **Entry** object that can be used to call a Connector's data access methods (.update(), delete(), etc.).

If you wish to proceed with the delete flow/operation, then you must set the current Entry with the following Connector function:

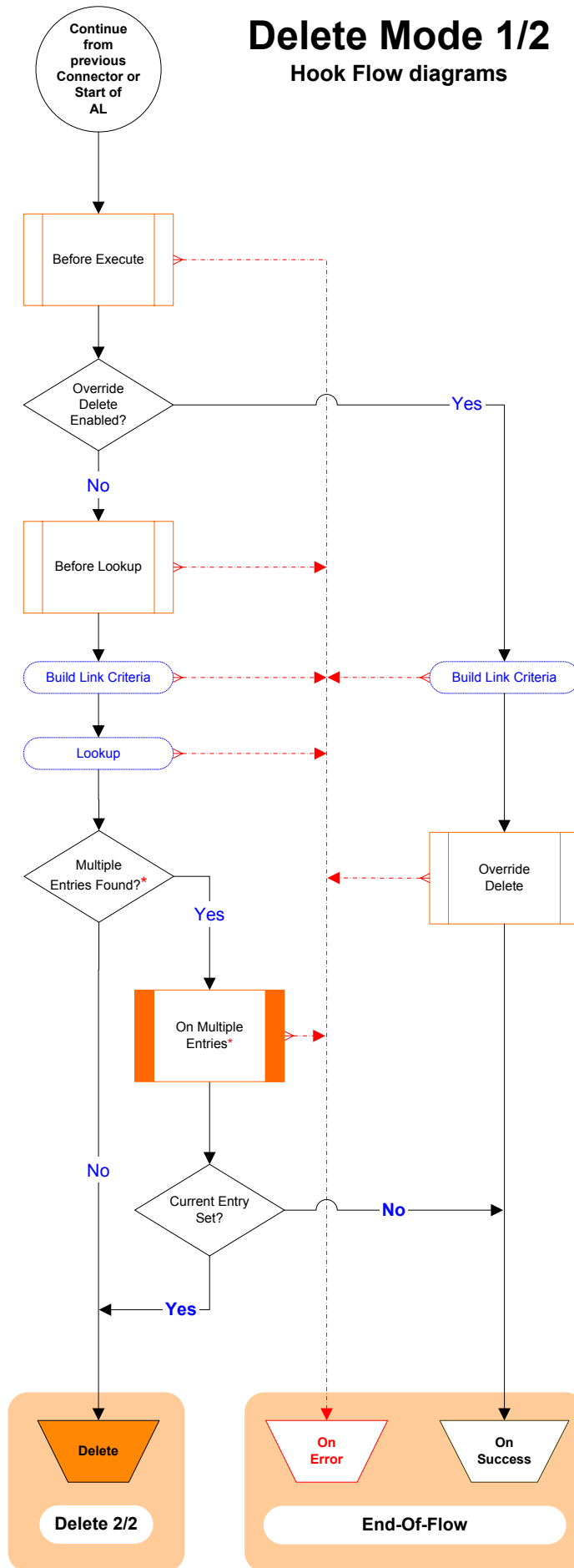
```
myConnector.setCurrent( myEntry )
```

If you do not set a current Entry, then execution will continue to On Success, bypassing the rest of the mode-specific flow.

Note:

Data sources behave differently when multiple Entries are to be handled.

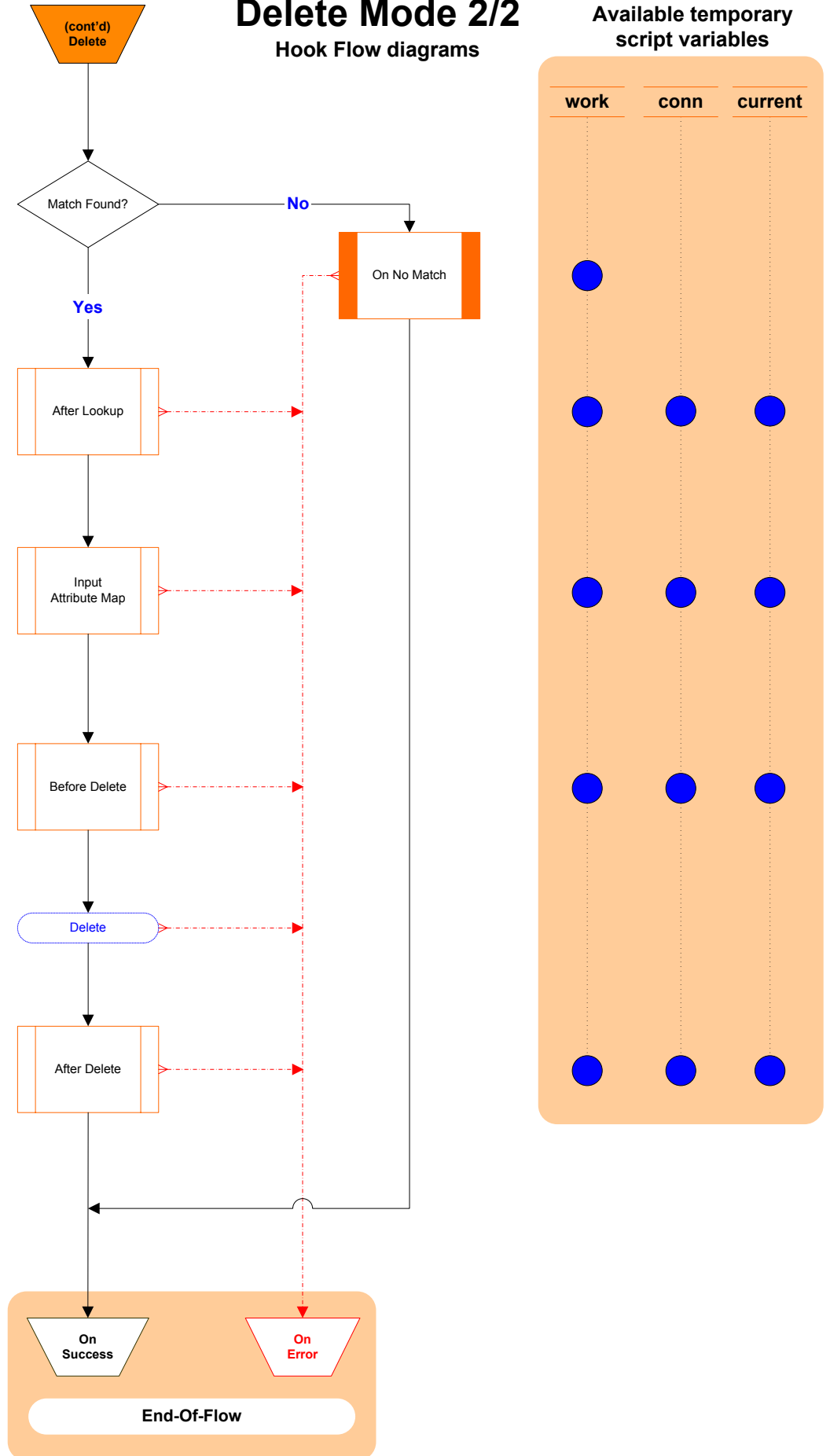
Even if you select a specific Entry as described above, it is not recommended that you continue with the delete flow, as this may result in an error, or that the operation is performed on multiple entries.



Delete Mode 2/2

Hook Flow diagrams

Available temporary script variables



Delta Mode 1/4

Hook Flow diagrams

Available temporary script variables

work

Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

After the **Build Link Criteria** operation, there is a script object called **search** available which gives you access to this information (e.g. for use in the Override Hook).

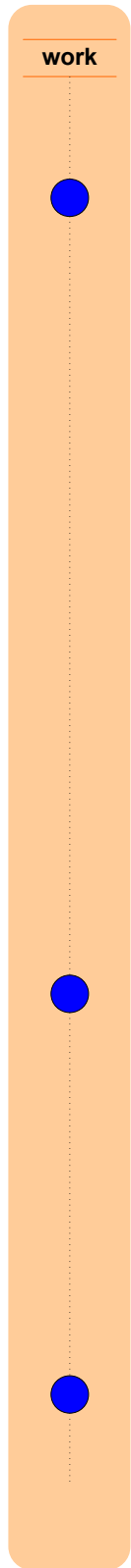
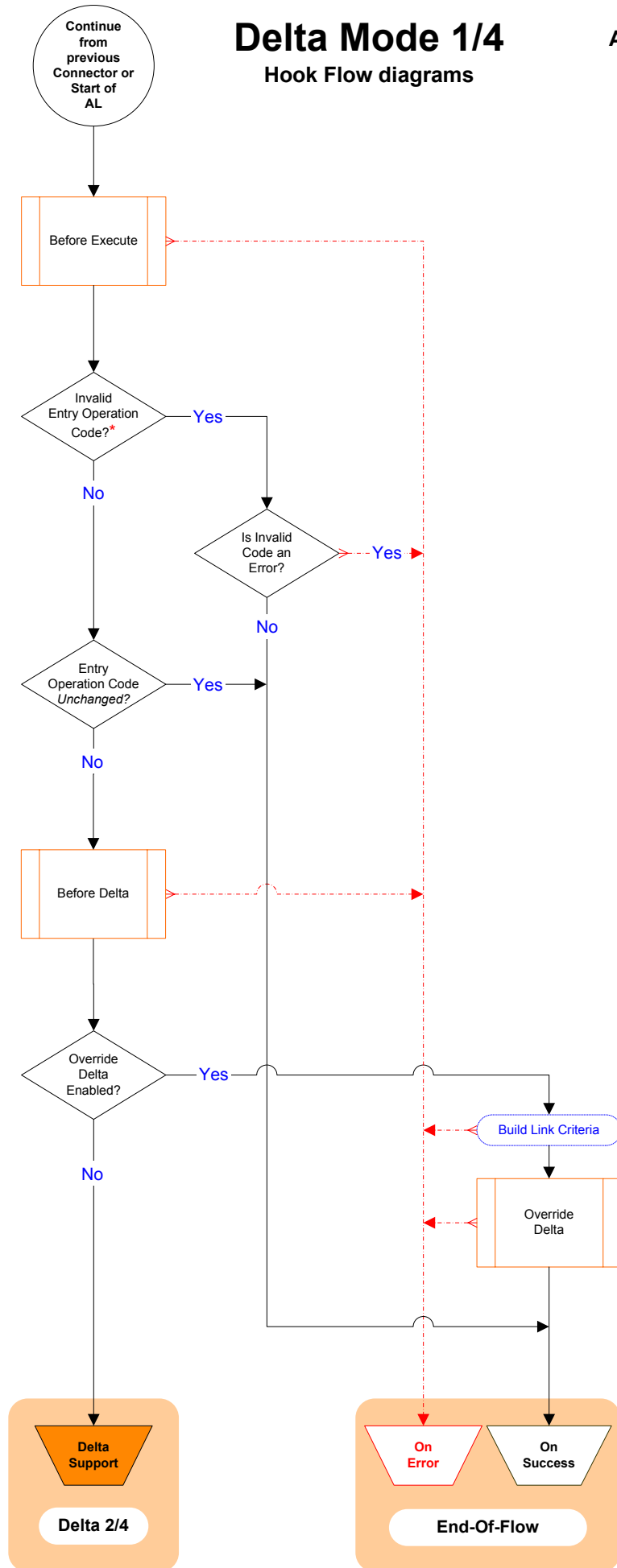
*** Valid Operation Code**

By default, an **exception** is thrown if Delta mode detects that the work Entry does not have a valid **operation code** (for example, "generic"). Operation code detection occurs after the **Before Execute** Hook. Delta mode can be configured to *ignore* these Entries instead.

Delta Application

During Delta processing, the necessary steps are taken to prepare for applying the detected changes as efficiently as possible.

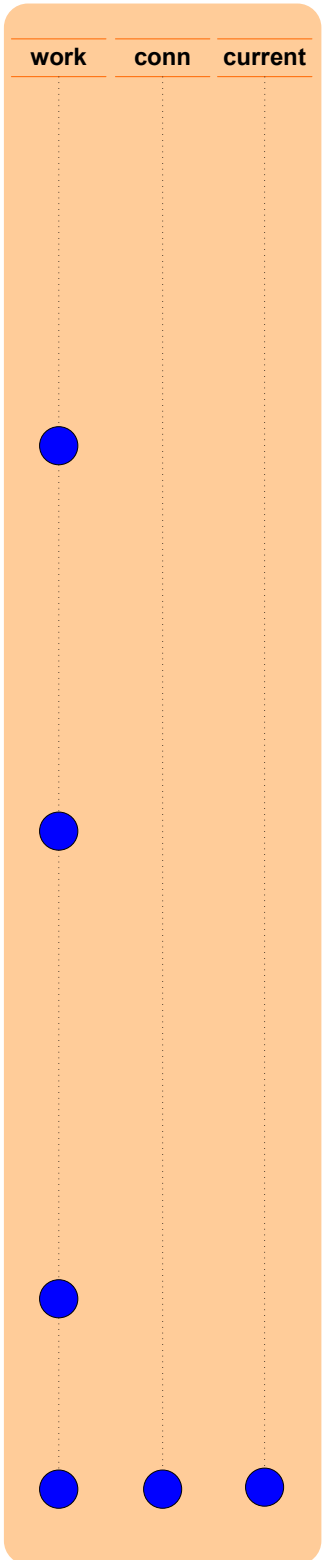
For example, **multi-value Attributes** require special handling so that **value-level Delta operation codes** are applied correctly.



Delta Mode 2/4

Hook Flow diagrams

Available temporary script variables



***On Multiple Entries**

If more than one record/entry is found that matches the Link Criteria then the On Multiple Entries Hook must also be **enabled**, or this is treated as an **error**.

You can access the set of records/entries found by using either of these two Connector functions:

```
getFirstDuplicateEntry()
or
getNextDuplicateEntry()
```

Each of these functions returns an **Entry** object that can be used to call a Connector's data access methods (.update(), delete(), etc.).

In addition, **conn** may be set to the desired **Entry** object by calling the Connector's **setCurrent()** function:

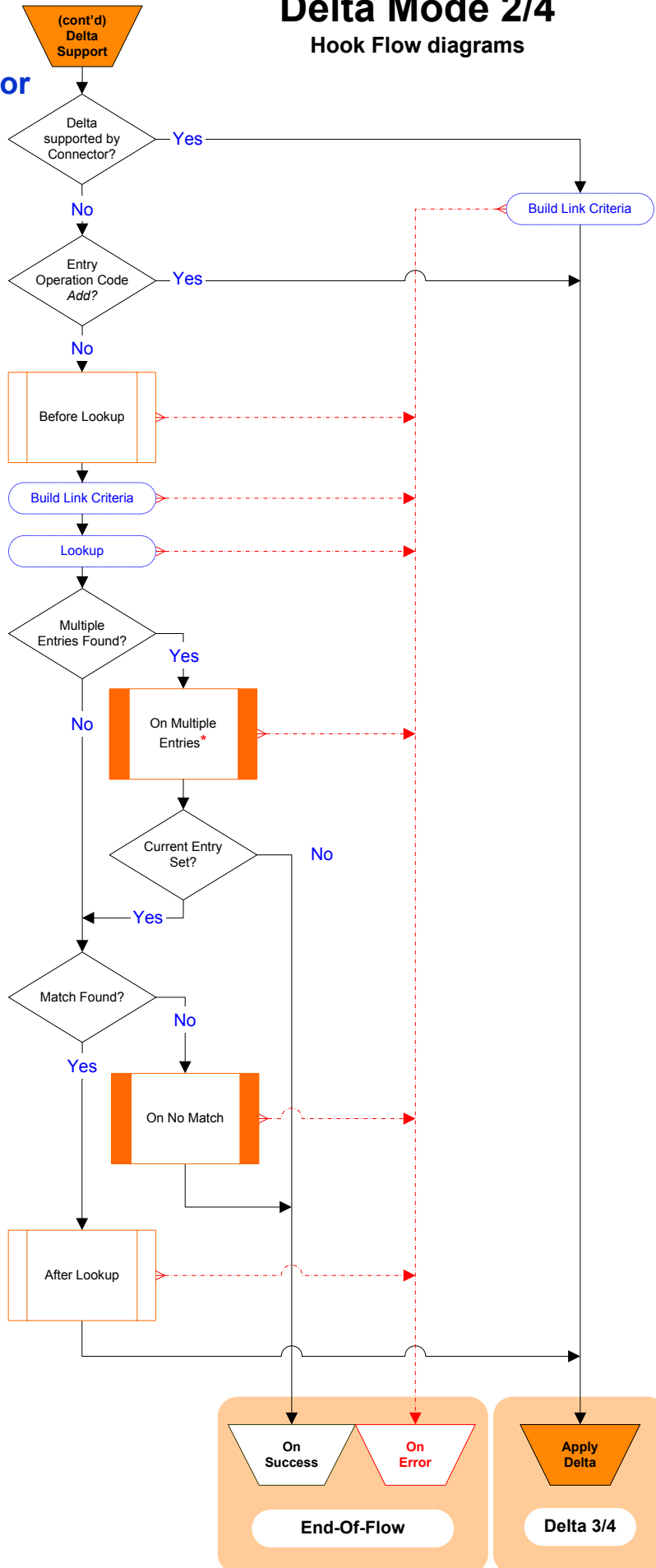
```
myConnector.setCurrent( myEntry )
```

If no Entry object is set, then execution will continue to **On Success**, skipping the rest of the mode-specific flow.

Note:

Please note that data sources (and therefore related Connectors) behave differently when multiple Entries are to be handled.

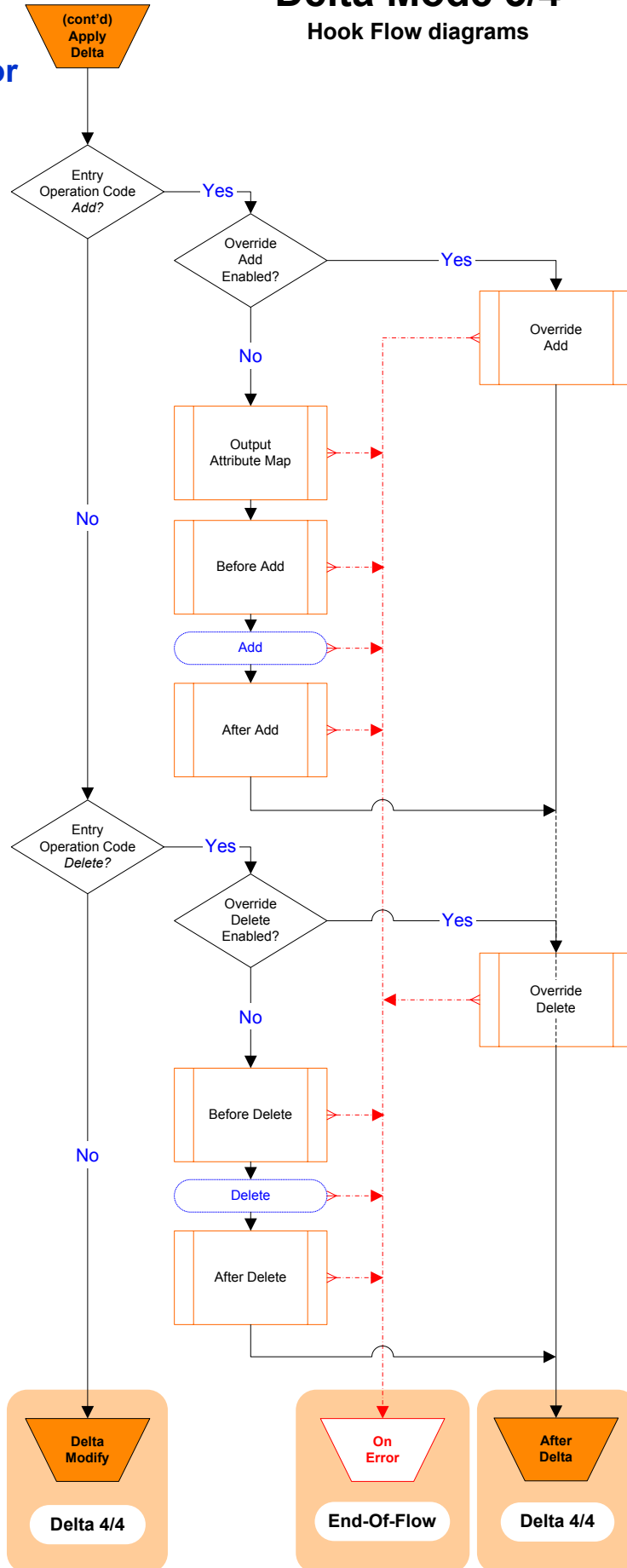
Even if you set a specific Entry as described above, it is not recommended that you continue with the delta operation, as this may result in an error, or that the operation is performed on multiple entries.



Delta Mode 3/4

Hook Flow diagrams

Available temporary script variables



work	conn	current
●		
●	●	
●	●	
●	●	
●	●	
●		
●	●	●
●	●	●
●	●	●

Delta Mode 4/4

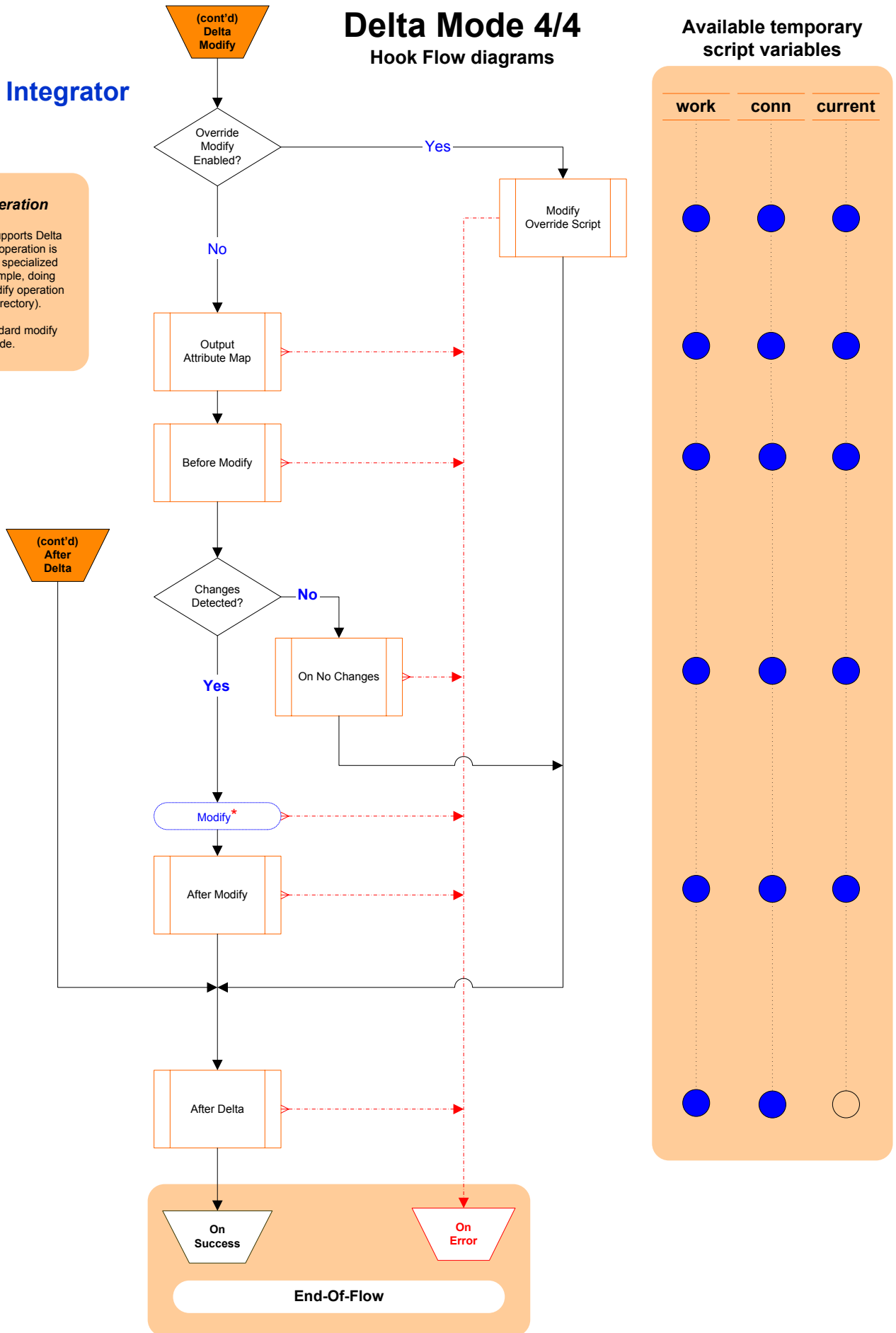
Hook Flow diagrams

Available temporary script variables

***Modify operation**

If the Connector supports Delta directly, then this operation is carried out by this specialized behavior (for example, doing an incremental modify operation for an LDAP directory).

Otherwise, a standard modify call is made.

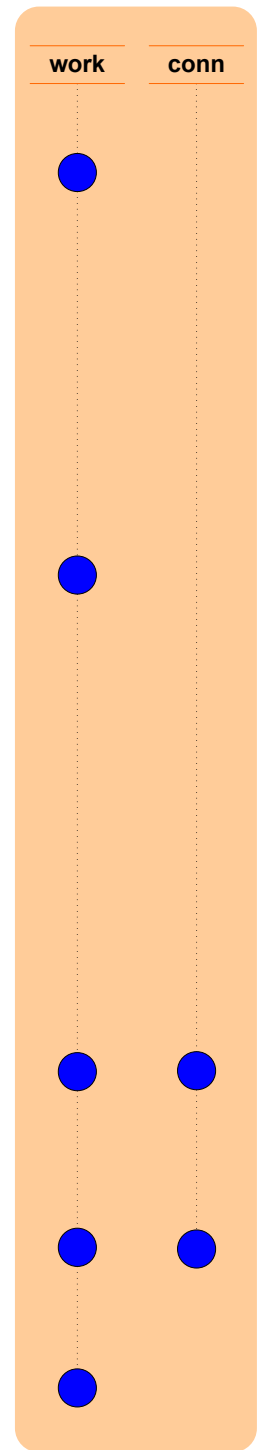


work	conn	current
●	●	●
●	●	●
●	●	●
●	●	●
●	●	●
●	●	●
●	●	○

Iterator Mode

Hook Flow diagrams

Available temporary script variables



Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

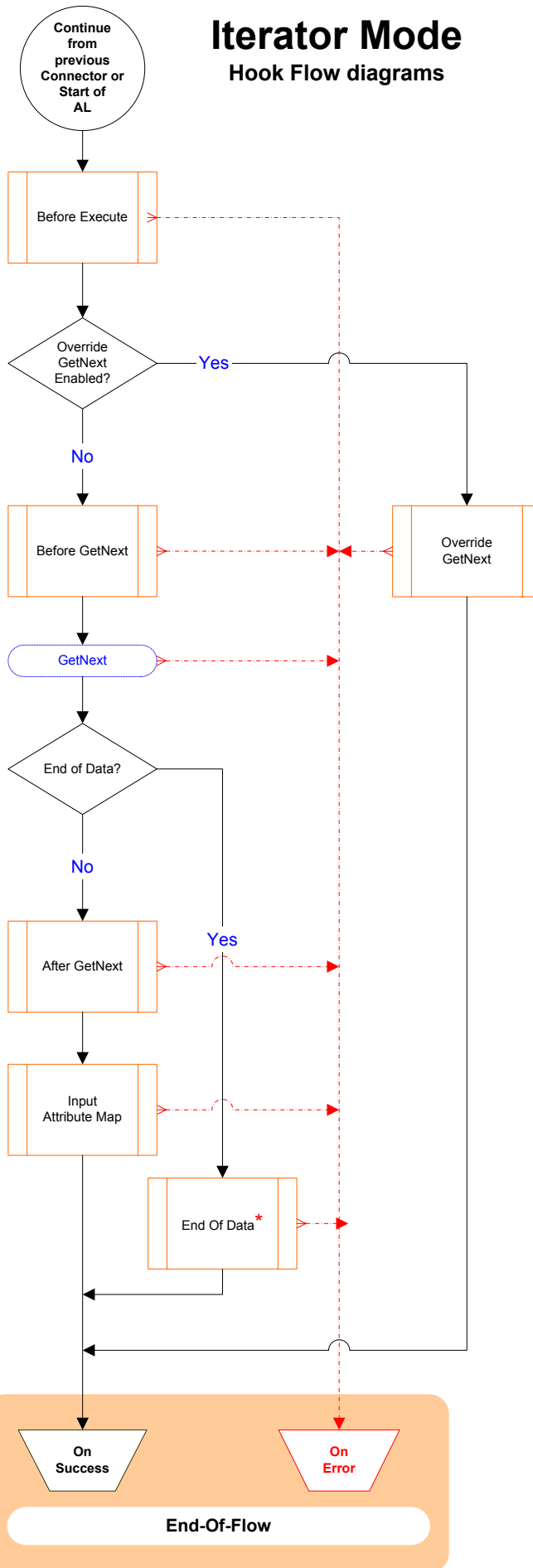
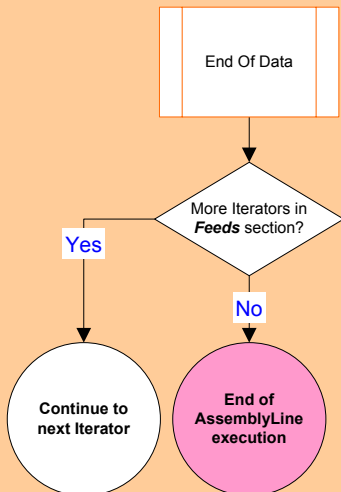
The data read in by each **GetNext** operation is available in the **conn** object.

Note:

If a Connector in Iterator mode detects the presence of a valid **work** object at the start of its execution - for example, that there is another Iterator in front of this one in the same AssemblyLine, or that the initial work Entry has been passed into the AssemblyLine from a calling process or system - then this Connector will not be executed, passing instead this Entry to the next Connector in the AssemblyLine.

The sidebar below illustrates what happens when an Iterator reaches its end-of-data. At this point it will not pass a work object to the next Connector, which in the case of another Iterator, will signal it to begin its own iteration.

***After the End Of Data** hook, execution flow continues as shown below:



Lookup Mode

Hook Flow diagrams

Available temporary script variables

work	conn	current
------	------	---------

Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

After the **Build Link Criteria** operation, there is a script object called **search** available which gives you access to this information (i.e. for use in the Override Hook).

The record/entry matching the **Link Criteria** is available through the **conn** object.

*On Multiple Entries

If more than one record/entry is found that matches the Link Criteria then the On Multiple Entries Hook must also be **enabled**, or this is treated as an **error**.

During this hook, **conn** may be set to the desired **Entry** object by calling the Connector's **setCurrent()** function:

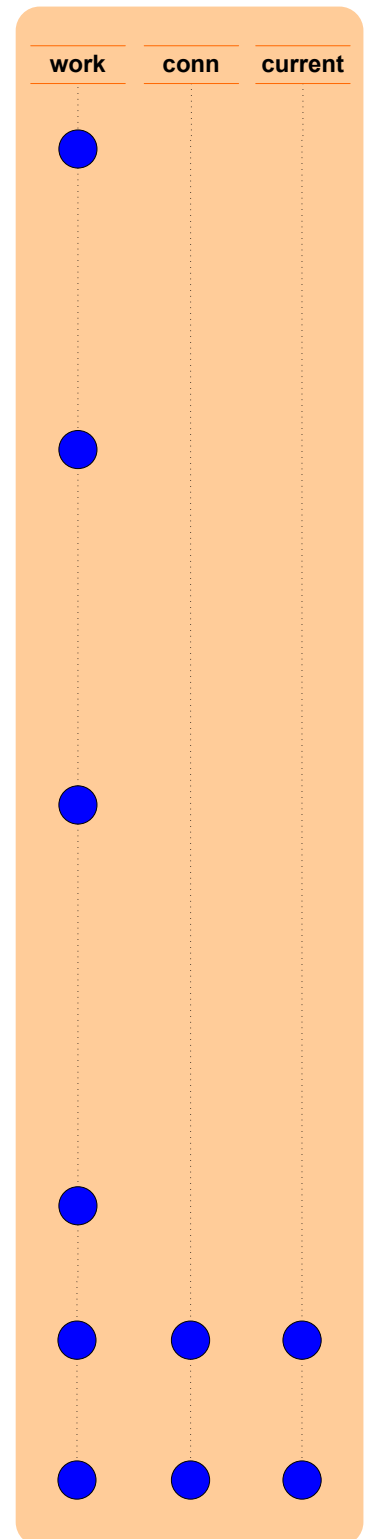
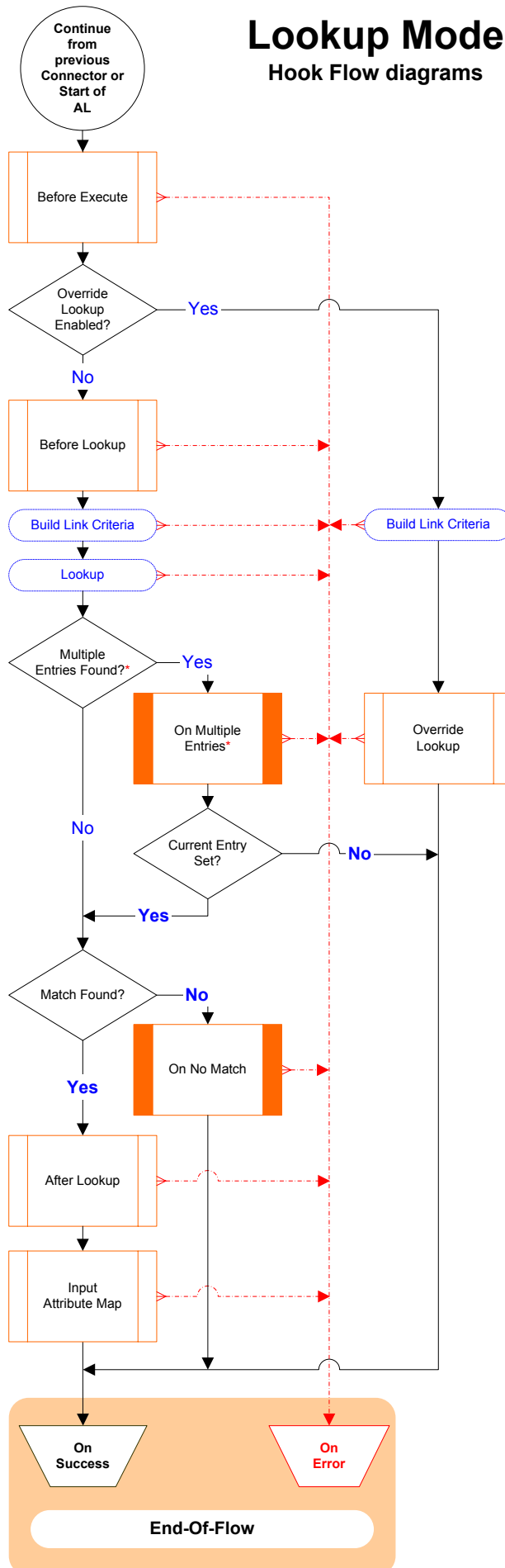
```
myConnector.setCurrent( myEntry )
```

You can access the set of records/entries found by using either of these two Connector functions:

```
getFirstDuplicateEntry()
or
getNextDuplicateEntry()
```

Each of these functions returns an **Entry** object that can be used in the **setCurrent()** call.

If **setCurrent()** is not called (e.g. no current entry is set) then the flow is passed on to **On Success**, skipping the rest of the mode-specific flow.



Server Mode

Hook Flow diagrams

Available temporary script variables

Available Objects

The only temporary Entry object is **conn**, which is available in the **After Accepting Connection** Hook.

This Entry contains a single Attribute called

connectorInterface

Its only value is a reference to the Connector Interface that will be paired up with the **Flow** component list in **Iterator Mode** to feed it with event data.

Server Behavior

Server Mode Connectors do *not* run exclusively like Iterators do. Instead, each is launched as a separate process in **event listening** mode and control is passed to the next **Feeds** Connector.

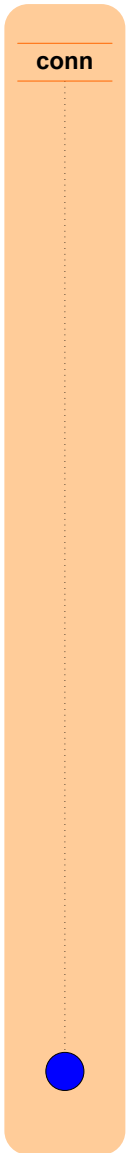
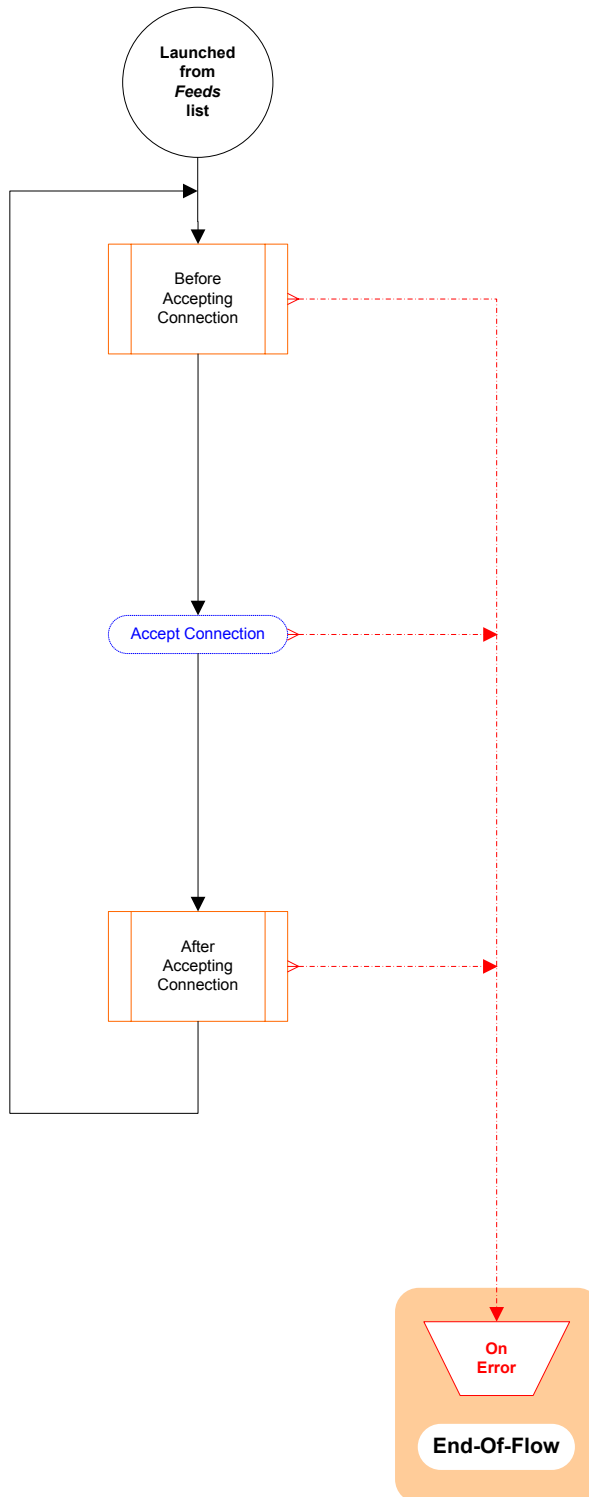
When an event is detected (for example, a client attempts to connect) then the Connector creates a **clone** of itself in **Iterator Mode** once the

After Accepting Connection Hook has completed.

This cloned Iterator is then paired up with the AssemblyLine **Flow** component list (possibly from the **AL Pool**) and Hook flow continues as with standard **Iterator** mode.

Furthermore, once the **Flow** section of the AssemblyLine completes, control is passed to the **Server Response** logic which then creates and sends the required reply to the caller/client system.

The Response Hook flow is detailed on the page entitled **Server Response**.



Available Objects

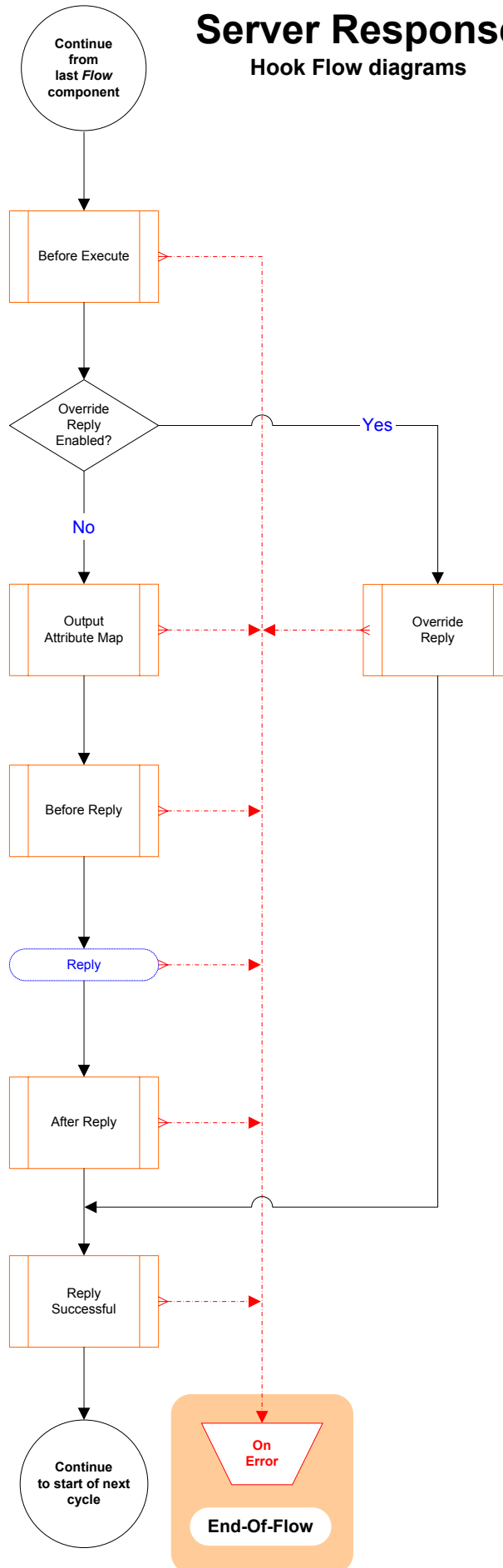
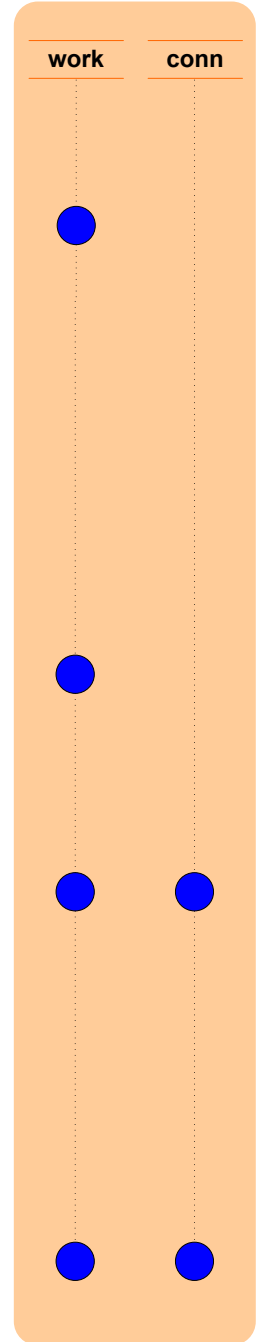
As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

The information stored in the **conn** object is sent to the data source by the **Reply** operation.

Server Response

Hook Flow diagrams

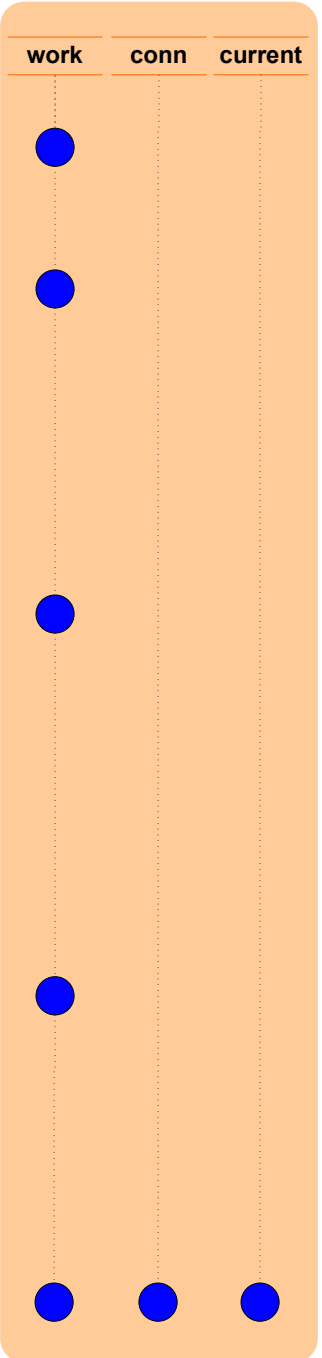
Available temporary script variables



Update Mode 1/3

Hook Flow diagrams

Available temporary script variables



Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

After the **Build Link Criteria** operation, there is a script object called **search** available which gives you access to this information (e.g. for use in the Override Hook).

***On Multiple Entries**

If more than one record/entry is found that matches the Link Criteria then the On Multiple Entries Hook must also be **enabled**, or this is treated as an **error**.

You can access the set of records/entries found by using either of these two Connector functions:

```
getFirstDuplicateEntry()
or
getNextDuplicateEntry()
```

Each of these functions returns an **Entry** object that can be used to call a Connector's data access methods (.update(), .delete(), etc.).

In addition, **conn** may be set to the desired **Entry** object by calling the Connector's **setCurrent()** function:

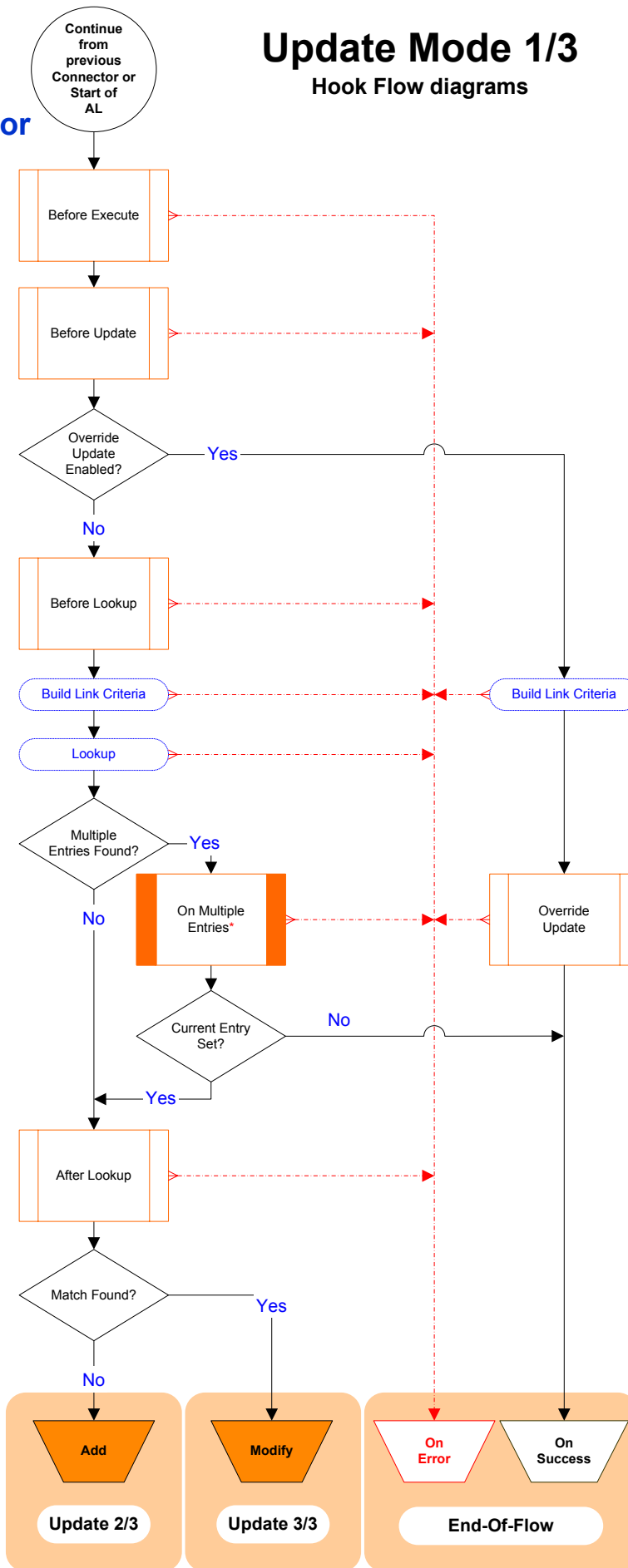
```
myConnector.setCurrent( myEntry )
```

If no Entry object is set, then execution will continue to **On Success**, skipping the rest of the mode-specific flow.

Note:

Please note that data sources (and therefore related Connectors) behave differently when multiple Entries are to be handled.

Even if you set a specific Entry as described above, it is not recommended that you continue with the update operation, as this may result in an error, or that the operation is performed on multiple entries.



Update Mode 2/3

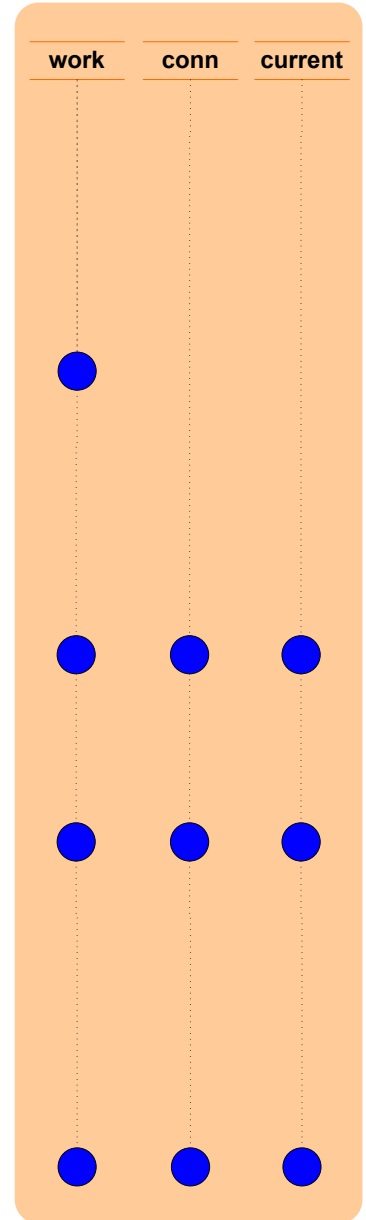
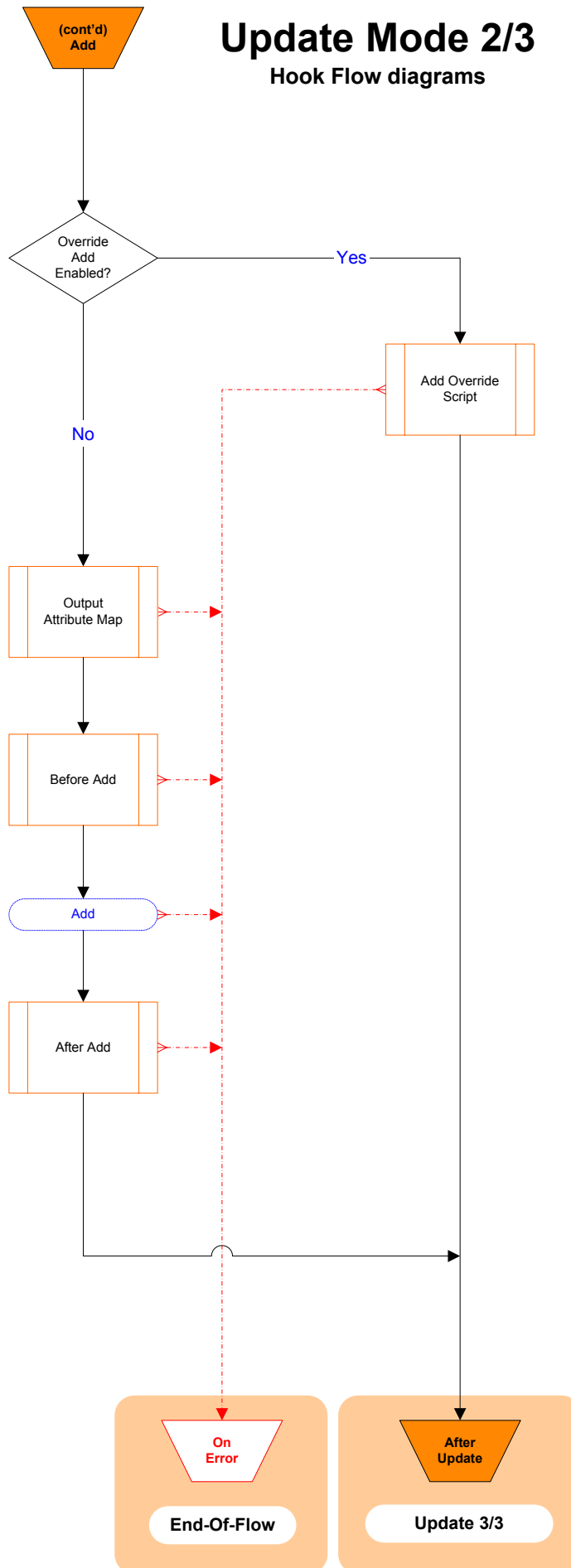
Hook Flow diagrams

Available temporary script variables

Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine.

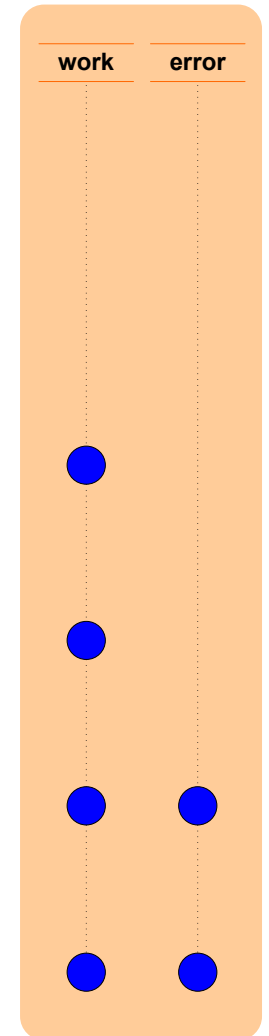
If the Update results in an **Add** operation, **conn** holds the data that is written to the data source.



End-Of-Flow for All Connector Modes

Hook Flow diagrams

Available temporary script variables



Available Objects

As always, **work** gives you access to the attributes that are currently in the AssemblyLine..

The **conn** and **curent** objects are available in the **On Error** and **On Success** Hooks if they were present previously in the flow

End-Of-Flow

This flow applies to all Connectors that either terminate normally (e.g. successfully) or due to an error.

The only exception is **Iterator** mode, which terminates the AssemblyLine if *end-of-data* is reached and there are no other Iterators to pass control to.

Error Handling

Please note that if either **On Error** Hook is enabled, then control is passed to the next Connector, as if the Connector had terminated successfully; Otherwise, the AssemblyLine **aborts**.

Furthermore, if an error occurs in an **On Error** Hook, then the AssemblyLine will also **abort**.

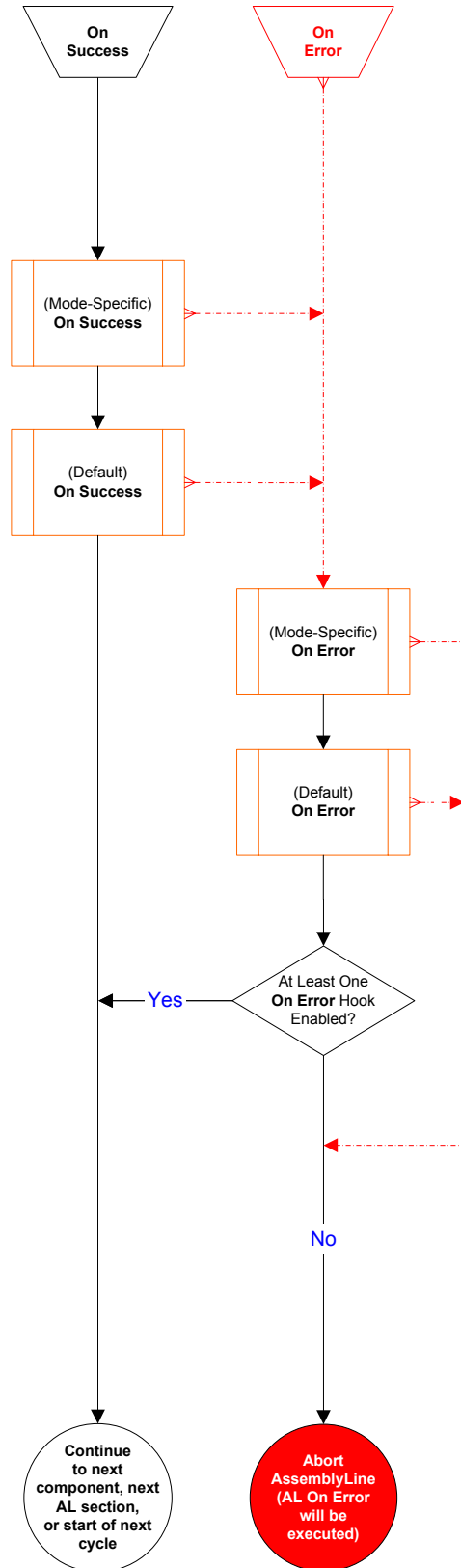
The **error** object (of type Entry) is available throughout an AssemblyLine, and provides information about the an error situation through its attributes: **status**, **exception**, **class**, **message**, **operation** and **connectormame**.

The **status** attribute will have the string value "OK" until an error situation arises, at which time it is assigned the value "fail" and the other attributes are added to **error**.

AssemblyLine End-of-Flow

If the AssemblyLine completes without unhandled errors, the AssemblyLine **On Success** Hook is invoked.

Otherwise, if an error has occurred than control is passed to the AssemblyLine **On Error** Hook.



Connector Reconnect

Hook Flow diagrams

Available Objects

As always, **work** gives you access to the attributes that are currently in the **AssemblyLine**.

The **error** object (of type **Entry**) is available throughout an **AssemblyLine**, and provides information about the an error situation through its attributes: **status**, **exception**, **class**, **message**, **operation** and **connectortname**.

The **status** attribute will have the string value "OK" until an error situation arises, at which time it is assigned the value "fail" and the other attributes are added to **error**.

*Auto Reconnect

The Auto Reconnect feature is configured through the parameters found in the Connector **Reconnect** tab.

These parameters control the maximum number of times a reconnect will be tried, as well as the number seconds to wait between each attempt.

