IBM.
Directory Integrator

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.

Continue from Previous Iterator or Start of AL

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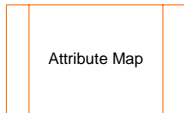
Attribute Map

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.

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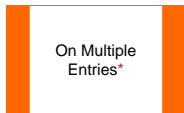
A few Hooks are *mandatory* and must be enabled, although they do not need to actually contain any script. If a mandatory Hook is not enabled and the flow reaches this point, then this is considered an **error**, and control faults out to error handling.

Lookup

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

Multiple Entries Found?* →Yes→
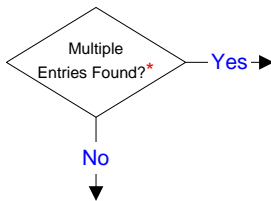
Decision components represent logical branches in component flow execution, depending on state information at this point.

No

For Each Non-Iterator

Mode-specific Flow*

(Enabled)
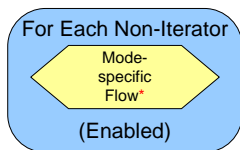
The yellow trapezoid describe flow which is detailed elsewhere (i.e. on another page in this document. The optional rounded blue box includes the

On Error

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.

Add

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

AssemblyLine Flow
Hook Flow diagrams

IBM.
Directory Integrator

*Flow References

These yellow trapezoids
represent flows
found in
the AssemblyLine
components.

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

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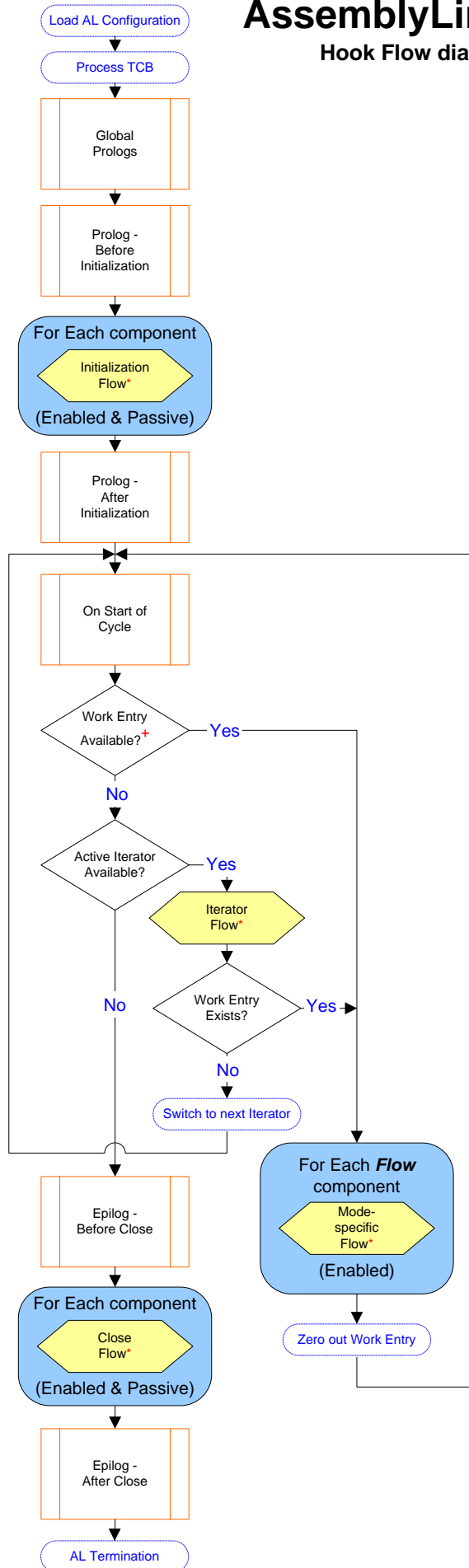described on the
page for **Iterator**
Mode flow.

Mode-specific Flow
can be found
on the page(s)
for that component
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()**

Load AL Configuration

Process TCB

Global
Prologs

Prolog -
Before
Initialization

For Each component
Initialization
Flow*
(Enabled & Passive)

Prolog -
After
Initialization

On Start of
Cycle

Work Entry
Available?+ → **Yes**

No

Active Iterator
Available? → **Yes**
Iterator
Flow*

Work Entry
Exists? → **Yes**

No (left branch)

No

Switch to next Iterator

**For Each *Flow*
component**
Mode-
specific
Flow*
(Enabled)

Zero out Work Entry

Epilog -
Before Close

For Each component
Close
Flow*
(Enabled & Passive)

Epilog -
After Close

AL Termination

Connector Initialization Flow

Hook Flow diagrams

Directory Integrator

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**.
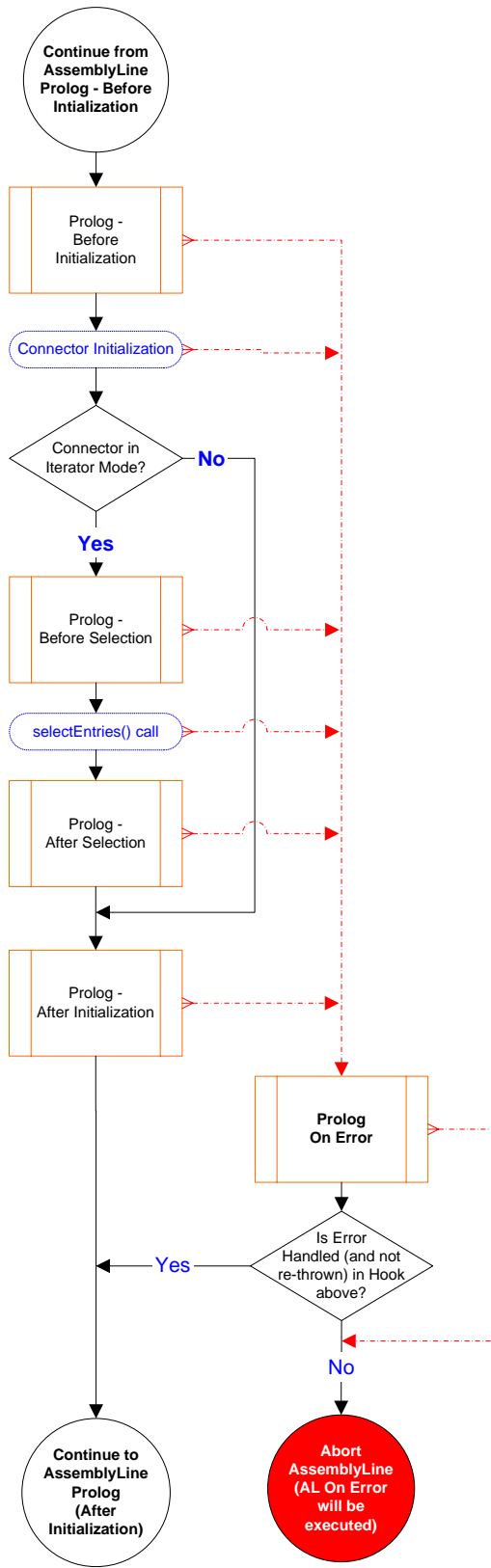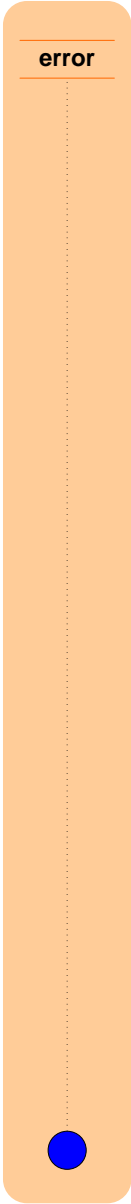
Error Handling

Please note that if the **Prolog On Error** Hook is enabled, then control is passed to back to the AssemblyLine flow; Otherwise, the AssemblyLine **aborts**.

The error condition can be passed on to next On Error Hook (i.e. to the AssemblyLine Error Hook) by re-throwing the exception:

throw error.getObject("exception");

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 error situation through its attributes: **status**, **exception**, **class**, **message**, **operation** and **connectorname**.

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**.

Available temporary script variables

error

Continue from AssemblyLine Prolog - Before Intialization

Prolog - Before Initialization

Connector Initialization

Connector in Iterator Mode? **No**

Yes

Prolog - Before Selection

selectEntries() call

Prolog - After Selection

Prolog - After Initialization

Prolog On Error

Is Error Handled (and not re-thrown) in Hook above?

Yes

No

Continue to AssemblyLine Prolog (After Initialization)

Abort AssemblyLine (AL On Error will be executed)

Connector
Close Flow
Hook Flow diagrams

IBM.
Directory Integrator

error

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.

Error Handling

Please note that if the **Prolog On
Error** Hook is enabled, then control
is passed to back to the
AssemblyLine flow;
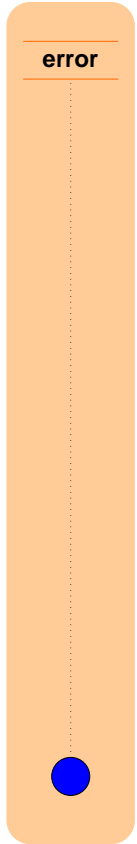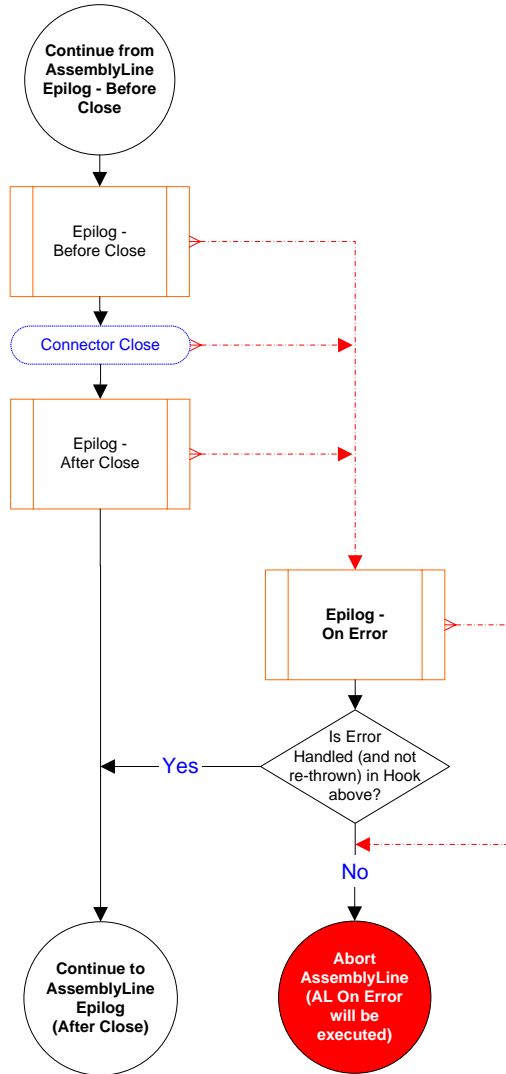Otherwise, the AssemblyLine
aborts.

The error condition can be
passed on to next On Error
Hook (i.e. to the
AssemblyLine Error Hook)
by re-throwing the exception:

throw error.getObject("exception");

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 error situation
through its attributes:
status, **exception**, **class**, **message**,
operation and **connectorname**.

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**.

**Continue from
AssemblyLine
Epilog - Before
Close**

Epilog -
Before Close

Connector Close

Epilog -
After Close

Epilog -
On Error

Is Error
Handled (and not
re-thrown) in Hook
above?

Yes

No

**Continue to
AssemblyLine
Epilog
(After Close)**

**Abort
AssemblyLine
(AL On Error
will be
executed)**

AddOnly Mode
Hook Flow diagrams

Directory Integrator

Continue from previous component or Start of AL

Before Execute

Override AddOnly Enabled?

Yes

No

Override AddOnly

Output Attribute Map

Before Add

Add

After Add

On Success

On Error

End-Of-Flow

Available temporary script variables

work　**conn**

IBM.
Directory Integrator

Continue from previous component or Start of AL

Call/Reply Mode
Hook Flow diagrams

Available temporary script variables
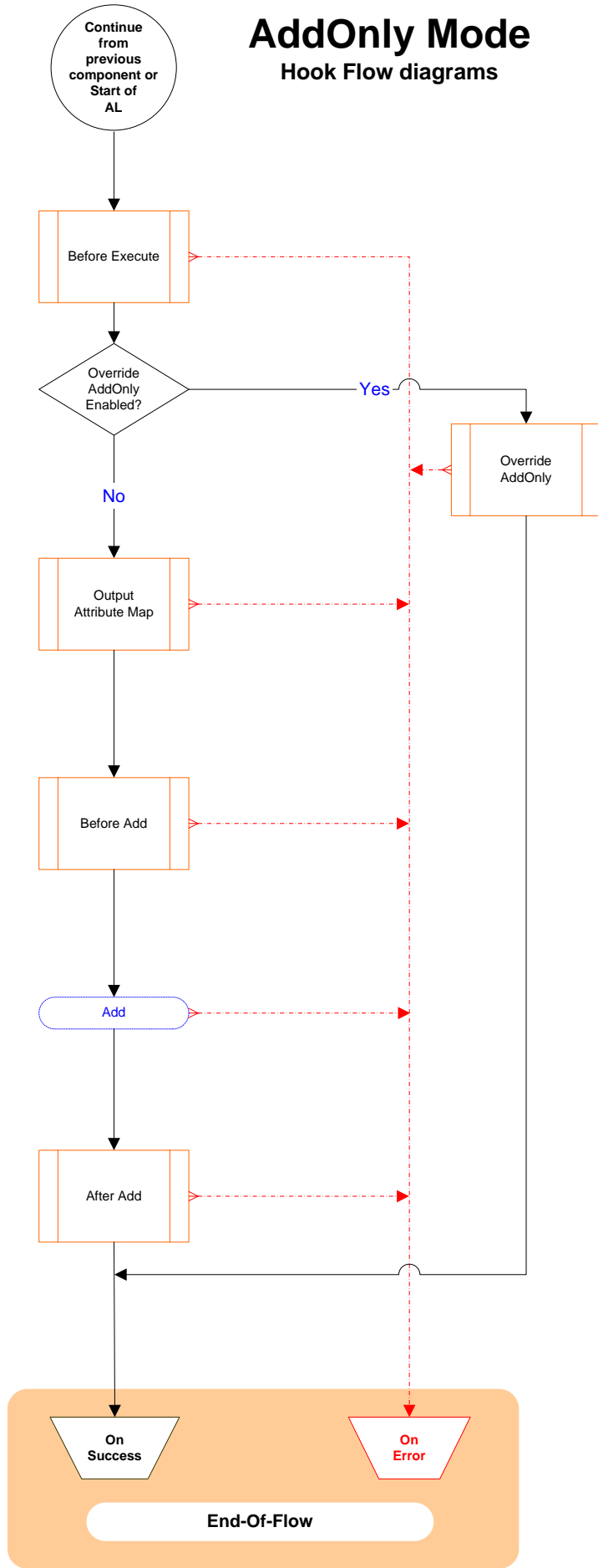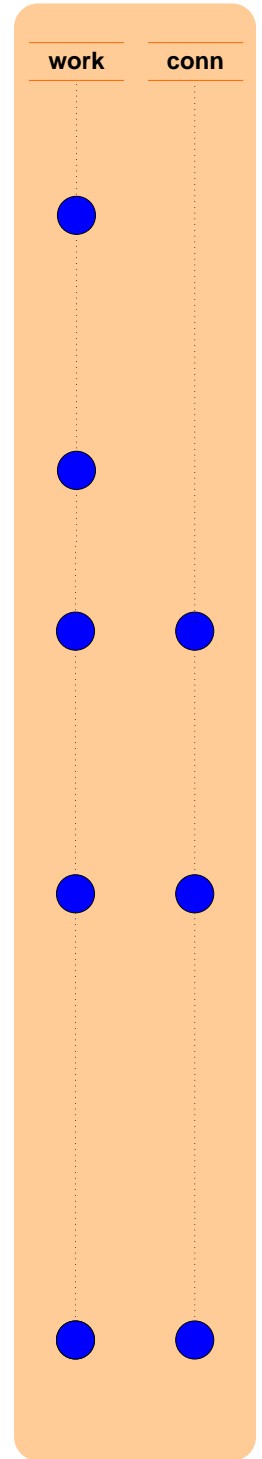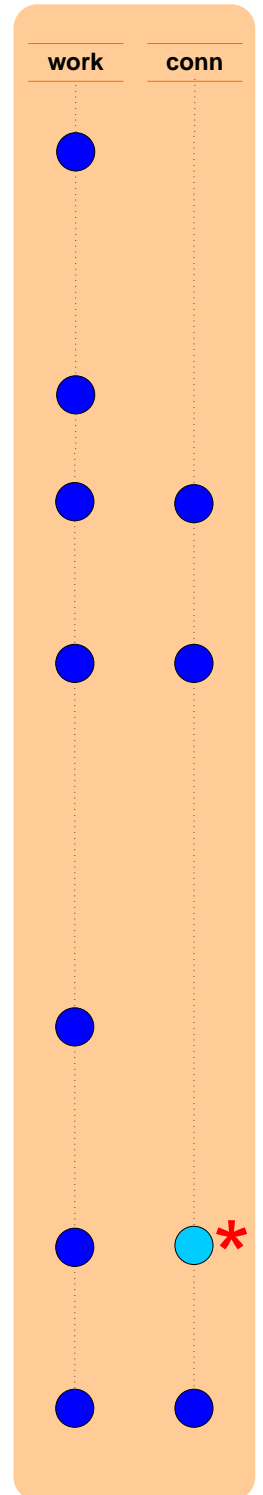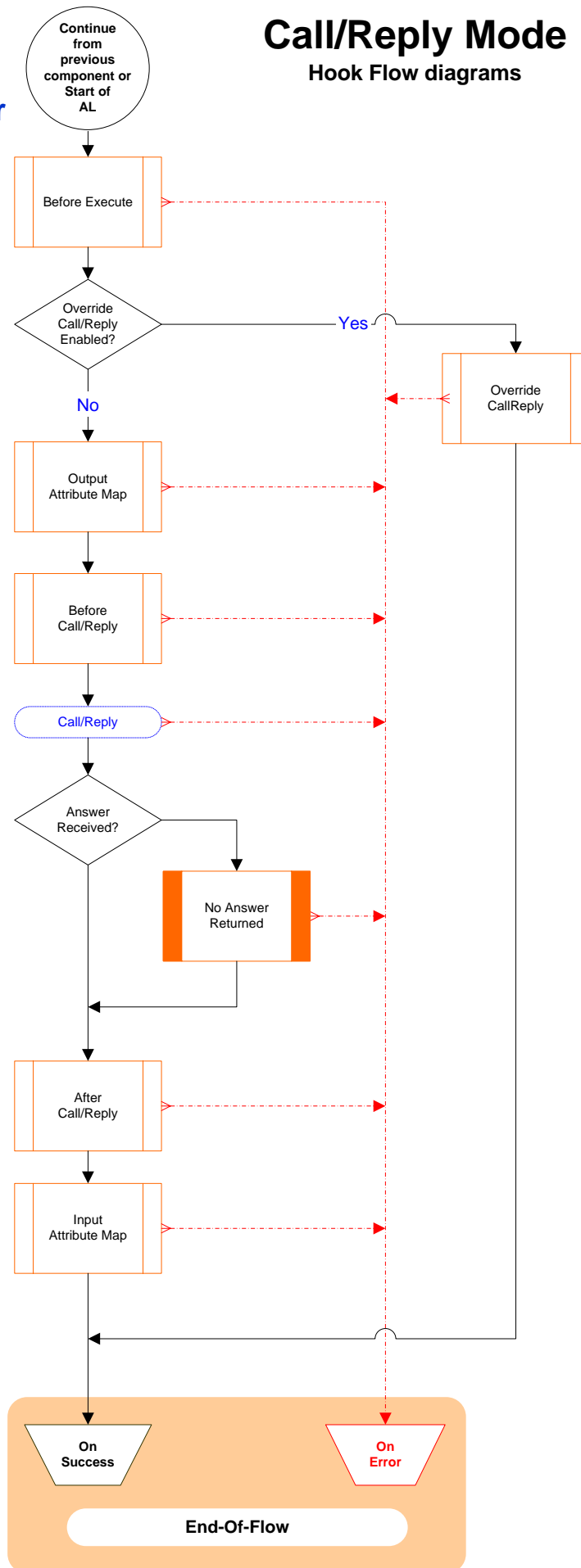
work   **conn**

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

Before Execute

Override Call/Reply Enabled?

Yes

No

Override CallReply

Output Attribute Map

Before Call/Reply

Call/Reply

Answer Received?

No Answer Returned

After Call/Reply

Input Attribute Map

On Success

On Error

End-Of-Flow

IBM. Directory Integrator

Delete Mode 1/2
Hook Flow diagrams
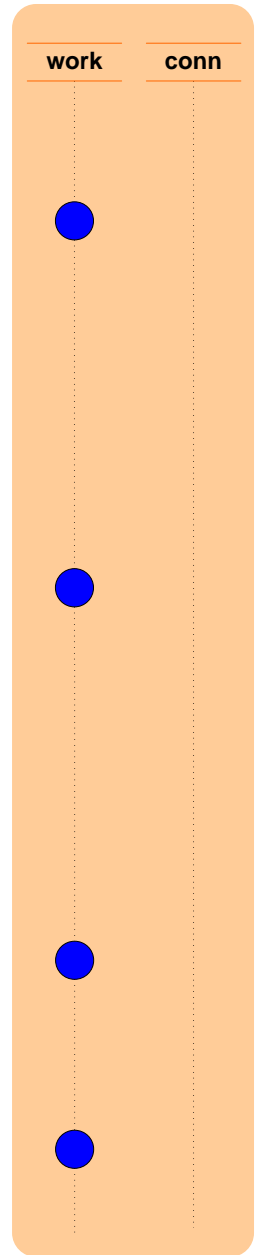
Available temporary script variables

work	conn

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 Interface'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:
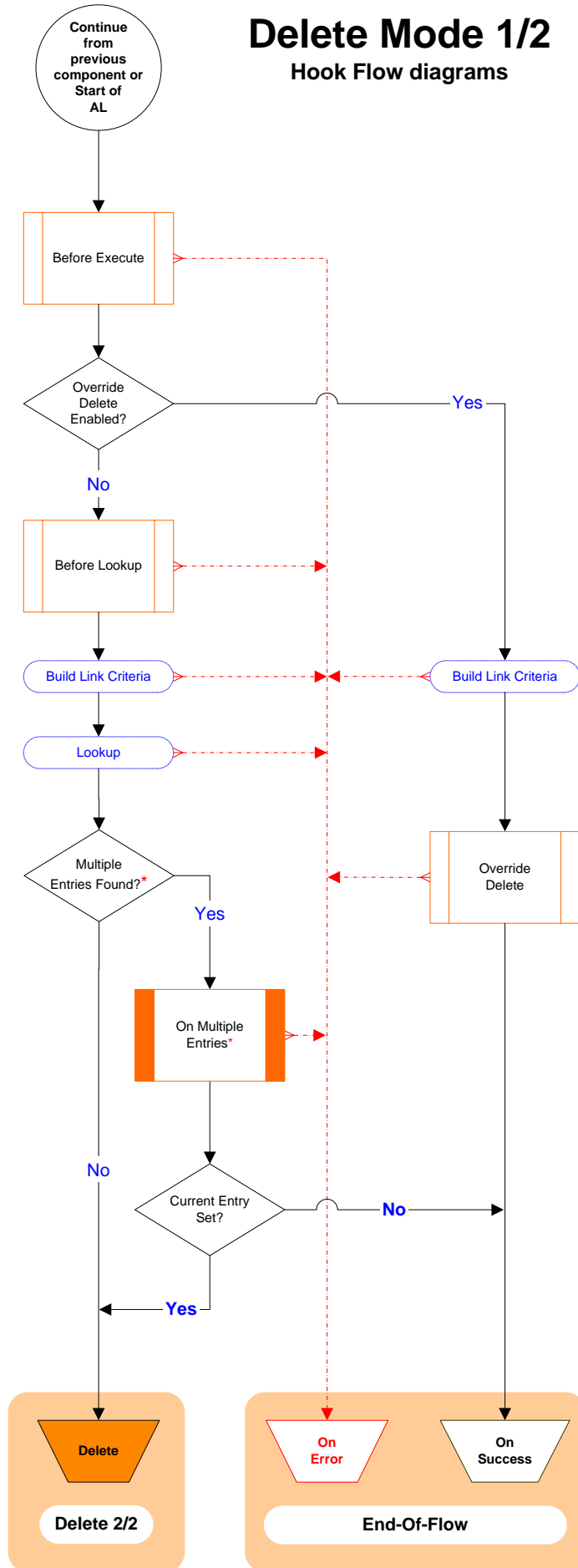
myConnector.setCurrent(*myEntry*)

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.

Continue from previous component or Start of AL

Before Execute

Override Delete Enabled? — Yes / No

Before Lookup

Build Link Criteria

Lookup

Build Link Criteria

Override Delete

Multiple Entries Found?* — Yes / No

On Multiple Entries*

Current Entry Set? — No / Yes

Delete

Delete 2/2

On Error

On Success

End-Of-Flow

IBM.

Directory Integrator

(cont'd) Delete

Delete Mode 2/2
Hook Flow diagrams

Available temporary script variables

work	conn	current

Match Found?

No

On No Match

Yes

After Lookup

Input Attribute Map

Before Delete

Delete

After Delete

On Success

On Error

End-Of-Flow
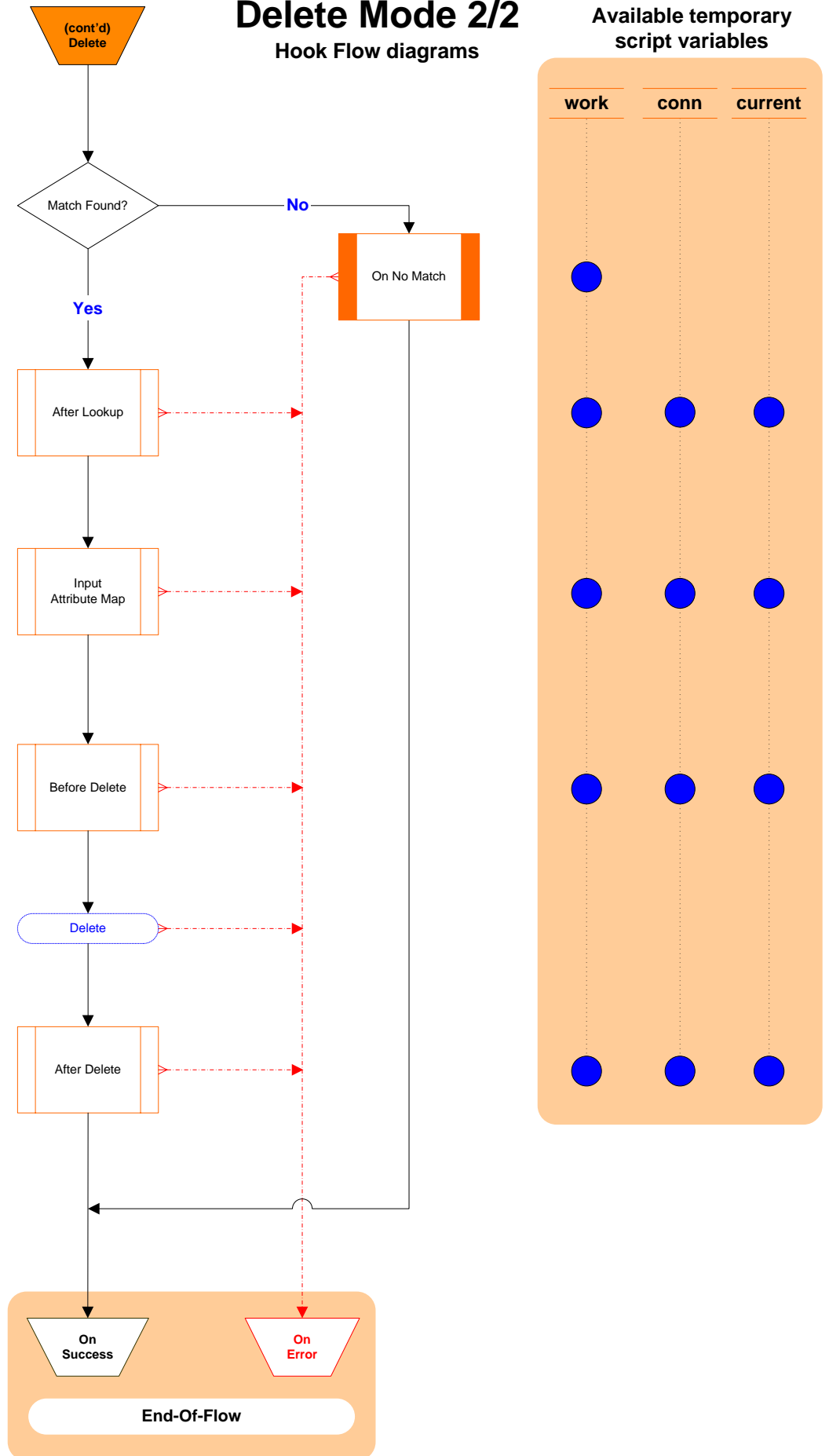
Delta Mode 1/4
Hook Flow diagrams

IBM. Directory Integrator

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).
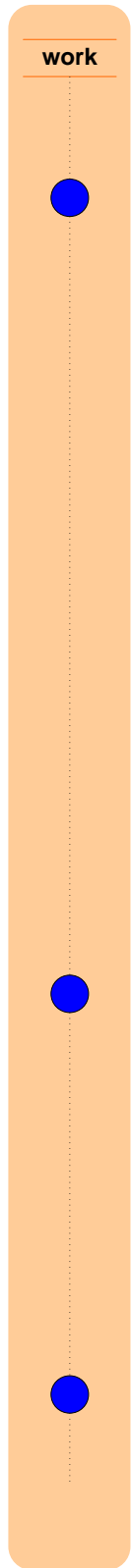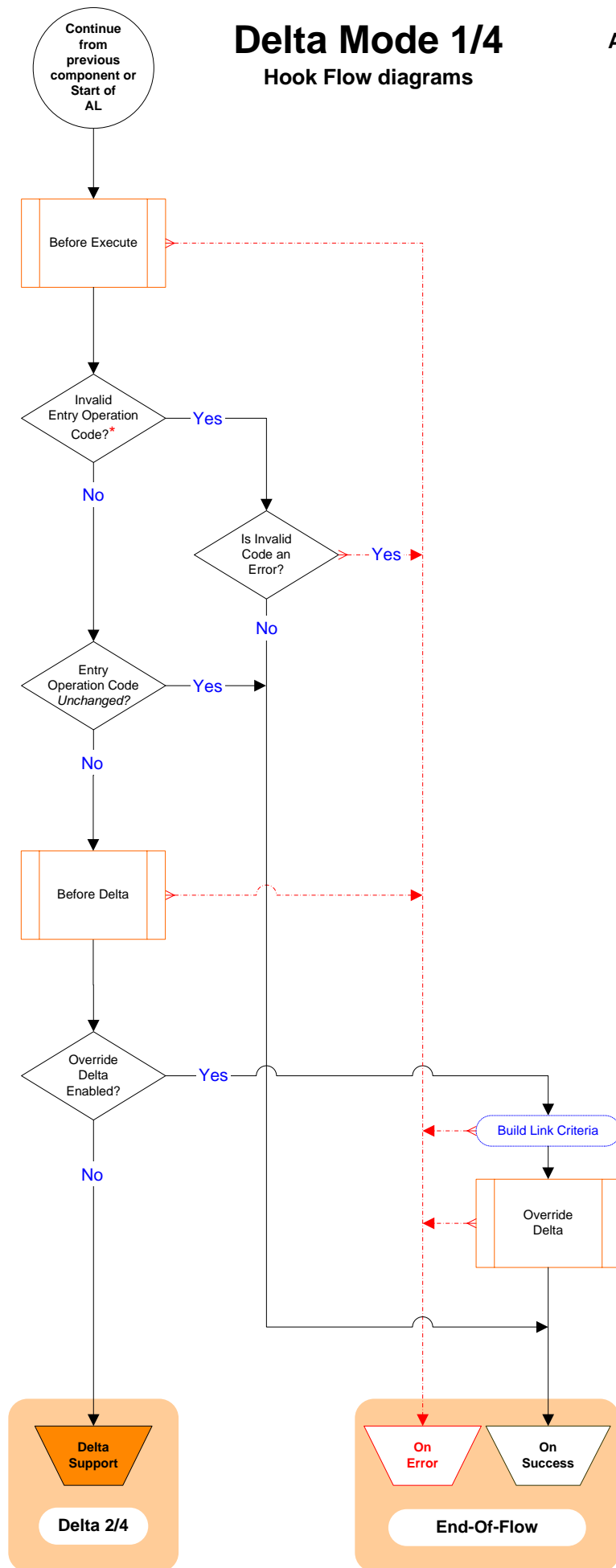
*Valid Operation Code

Be 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 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.

Continue from previous component or Start of AL

Before Execute

Invalid Entry Operation Code?* — Yes

No

Is Invalid Code an Error? — Yes

No

Entry Operation Code *Unchanged*? — Yes

No

Before Delta

Override Delta Enabled? — Yes

No

Build Link Criteria

Override Delta

Delta Support

Delta 2/4

On Error

On Success

End-Of-Flow

Hook Flow rev. 6.1
20060519

Delta Mode 2/4
Hook Flow diagrams

IBM
Directory Integrator

(cont'd) Delta Support

Available temporary script variables

work	conn	current

#Incr. Mod. possible?

The Connector checks to see if the underlying system supports incremental modifications.

For the LDAP Connector, this will always be Yes.

For the JDBC Connector the answer is currently always be No.

Incremental Modification possible?**#** — Yes

No

Build Link Criteria

Entry Operation Code *Add?* — Yes

No

Before Lookup

Build Link Criteria

Lookup

Multiple Entries Found? — Yes

No

On Multiple Entries*

Current Entry Set? — No

Yes

Match Found? — No

Yes

On No Match

After Lookup

*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 Interface'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:
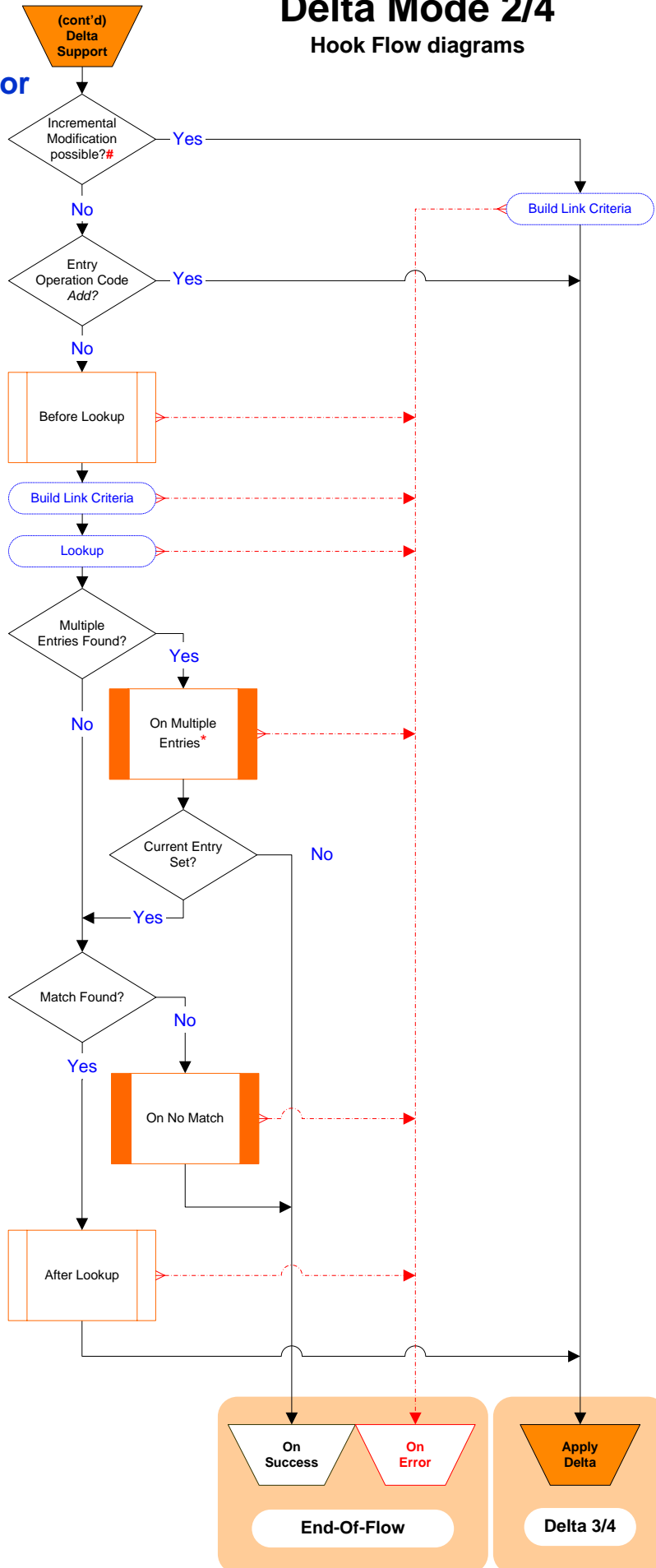
myConnector.setCurrent(*myEntry*)

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.
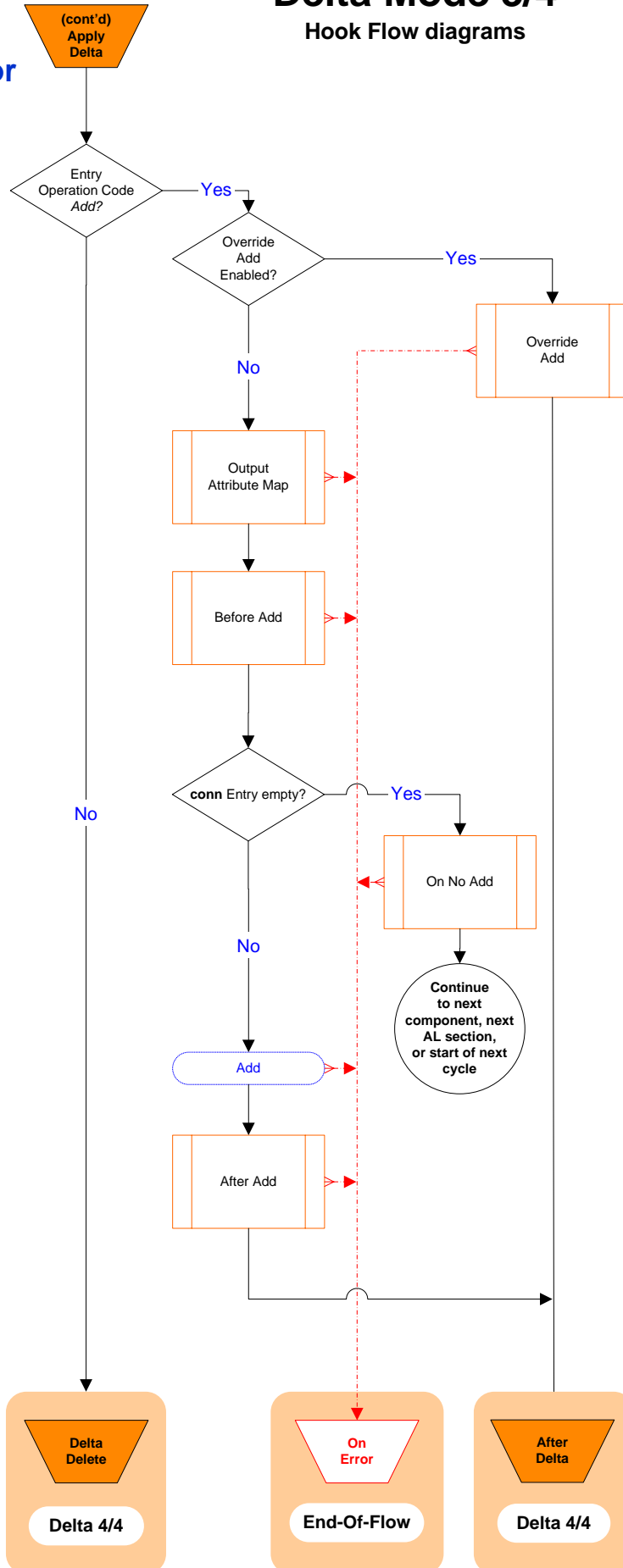
On Success   **On Error**

Apply Delta

End-Of-Flow   **Delta 3/4**

Hook Flow rev. 6.1
20060519

IBM.
Directory Integrator

(cont'd) Apply Delta

Delta Mode 3/4
Hook Flow diagrams

Available temporary script variables

work	conn	current

Entry Operation Code *Add?* — Yes

No

Override Add Enabled? — Yes

No

Override Add

Output Attribute Map

Before Add

conn Entry empty? — Yes

No

On No Add

Continue to next component, next AL section, or start of next cycle

Add

After Add

Delta Delete
Delta 4/4

On Error
End-Of-Flow

After Delta
Delta 4/4

Hook Flow rev. 6.1
20060519

Delta Mode 4/4
Hook Flow diagrams

IBM.
Directory Integrator

(cont'd) Delta Delete

Available temporary script variables

work	conn	current

*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.

Entry Operation Code *Delete?* — Yes

Override Delete Enabled? — Yes

No

Before Delete

Delete

After Delete

Override Delete

Override Modify Enabled? — Yes

No

Modify Override Script

Output Attribute Map

Before Modify

(cont'd) After Delta

Changes Detected? — **No**

Yes

On No Changes

Modify*

After Modify

After Delta

On Success

On Error

End-Of-Flow

Hook Flow rev. 6.1
20060519
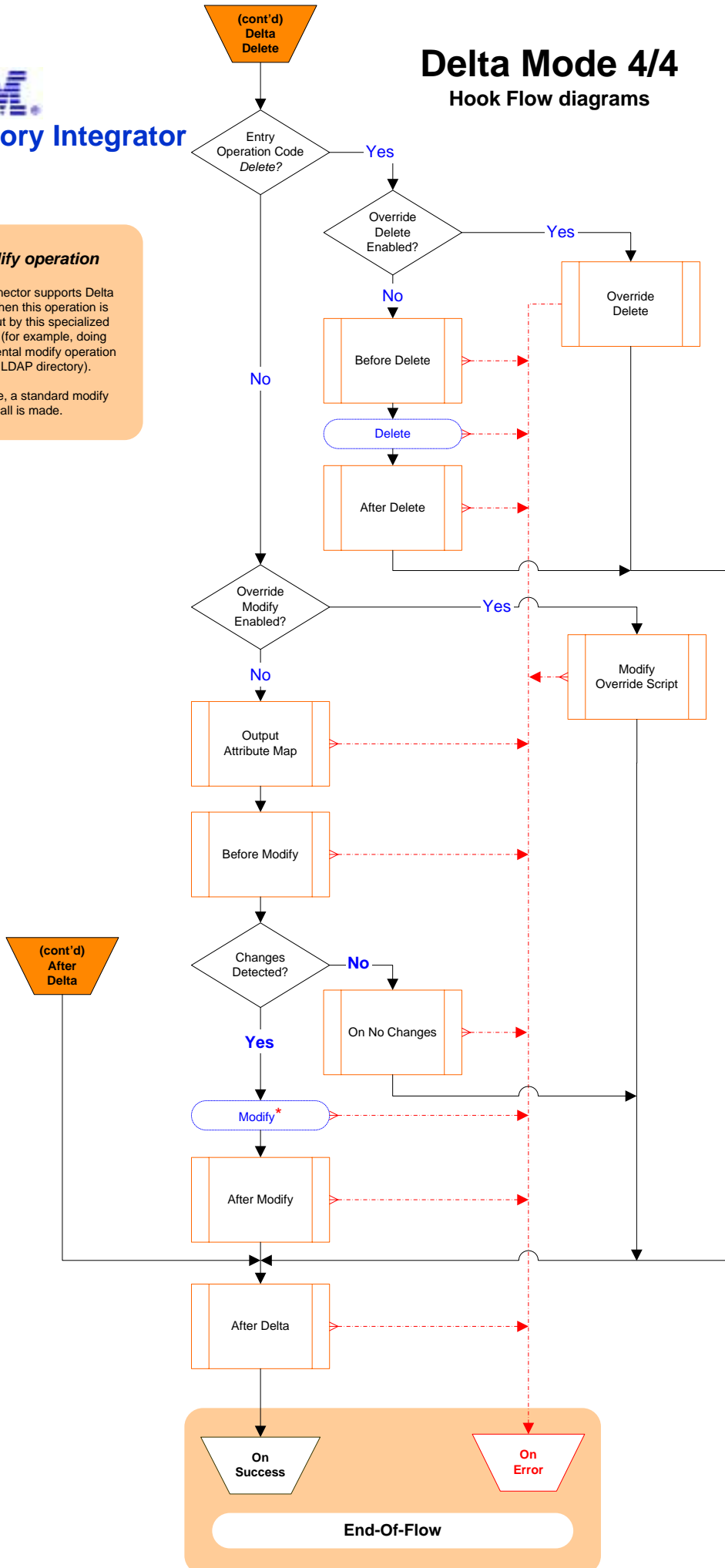
Iterator Mode
Hook Flow diagrams

**Available temporary
script variables**

work   conn

Continue
from
previous
component or
Start of
AL

Before Execute

Override
GetNext
Enabled?   Yes

No

Before GetNext   Override
GetNext

GetNext

End of Data?

No

After GetNext   Yes

Input
Attribute Map

End Of Data *

On
Success

On
Error

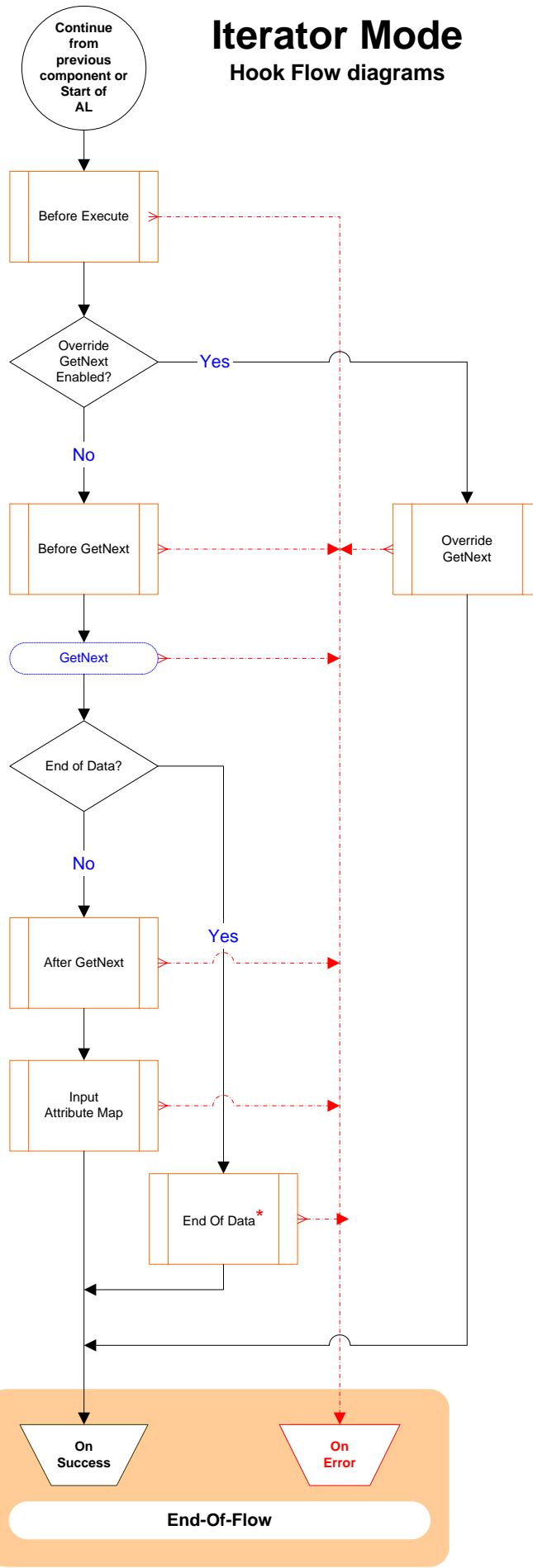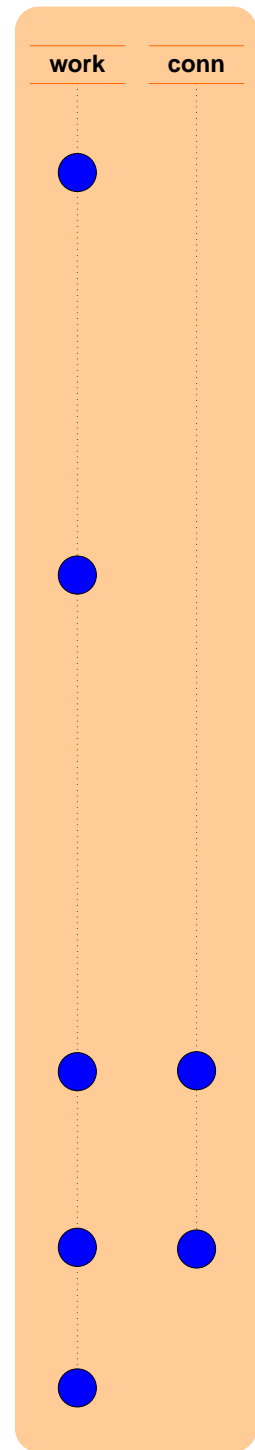**End-Of-Flow**

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
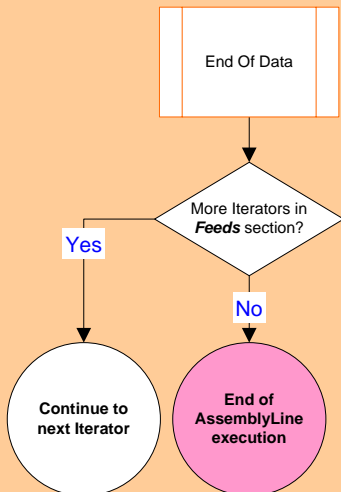available in the **conn** object.

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**
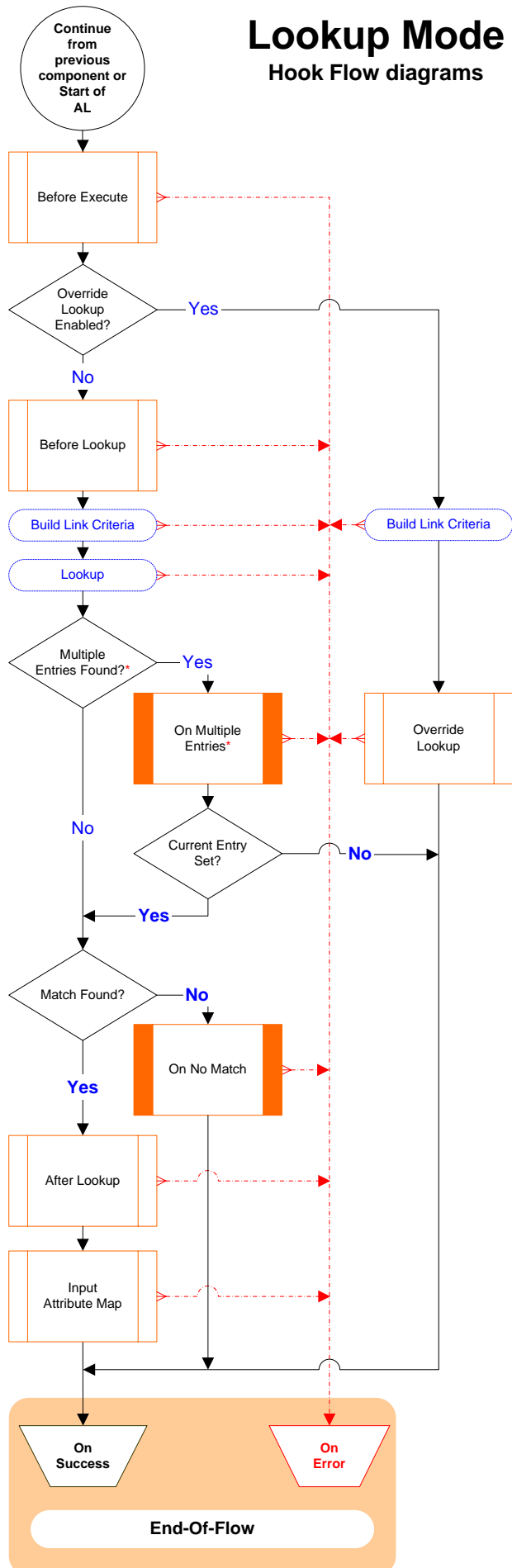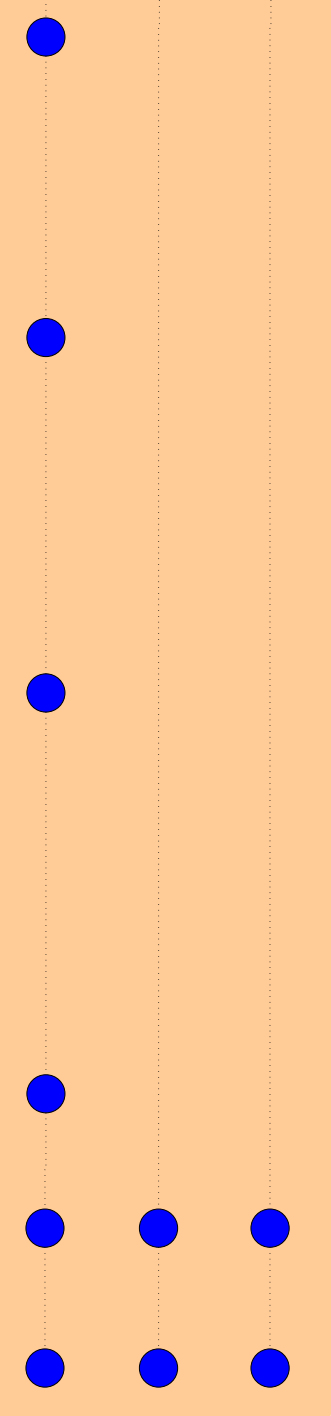hook, execution flow continues
as shown below:

End Of Data

More Iterators in
Feeds section?

Yes

No

Continue to
next Iterator

End of
AssemblyLine
execution

Directory Integrator

Lookup Mode
Hook Flow diagrams

Continue from previous component or Start of AL

Before Execute

Override Lookup Enabled? — **Yes**

No

Before Lookup

Build Link Criteria ········· Build Link Criteria

Lookup

Multiple Entries Found?* — **Yes**

On Multiple Entries* ····· Override Lookup

No

Current Entry Set? — **No**

Yes

Match Found? — **No**

Yes

On No Match

After Lookup

Input Attribute Map

On Success

On Error

End-Of-Flow

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.

Available temporary script variables

work	conn	current

Server Mode
Hook Flow diagrams

IBM.
Directory Integrator

Available temporary script variables

conn

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 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.
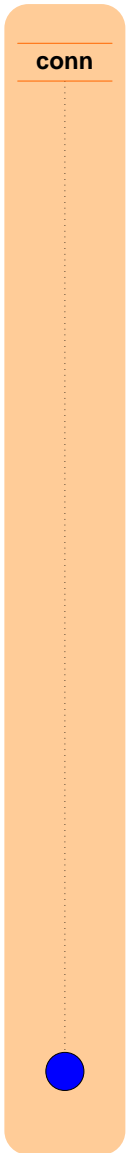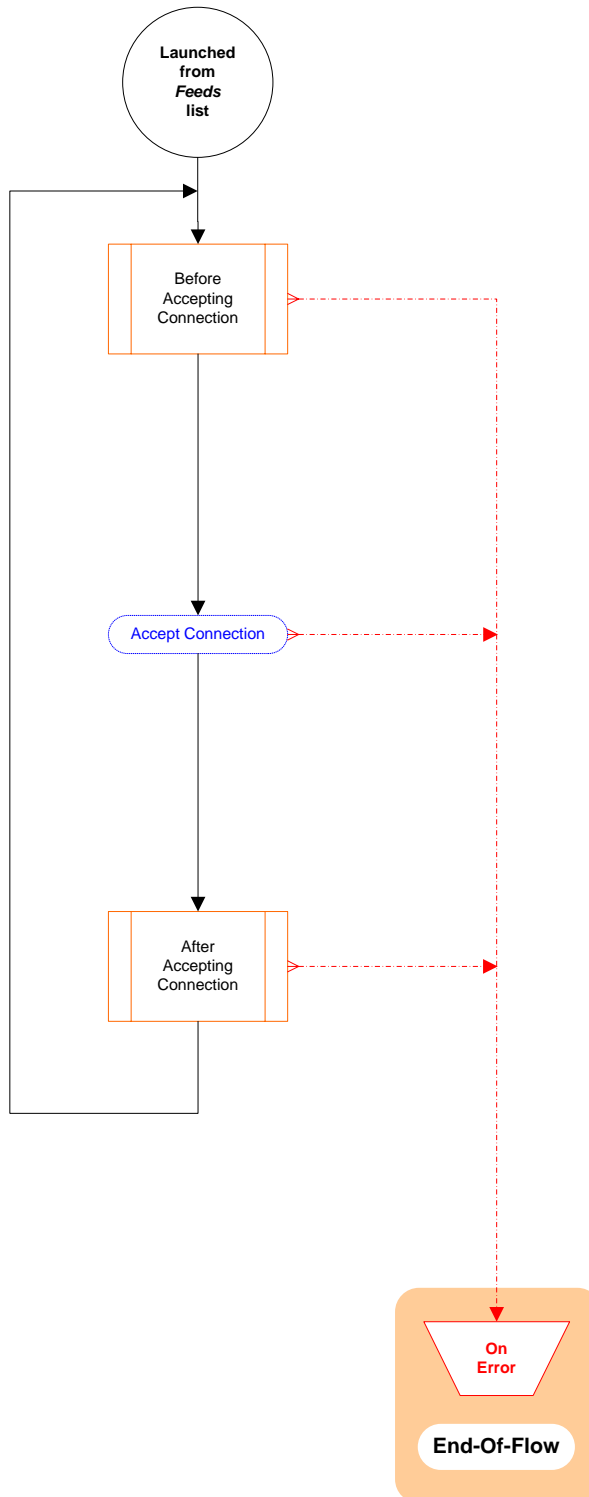
When an event is detected (for example, a client attempts to connect) then the Connector creates a **clone** of itself in **Iterator** Mode once the **After Accepting Connection** Hook has completed.

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**.

Launched from *Feeds* list

Before Accepting Connection

Accept Connection

After Accepting Connection

On Error

End-Of-Flow
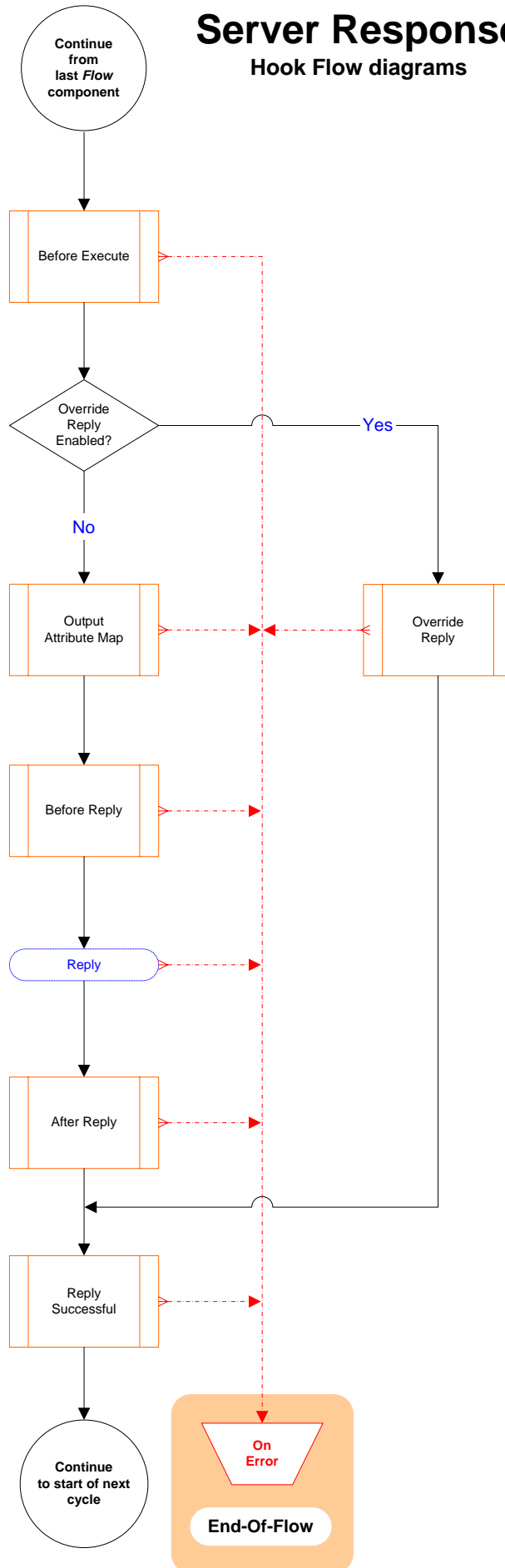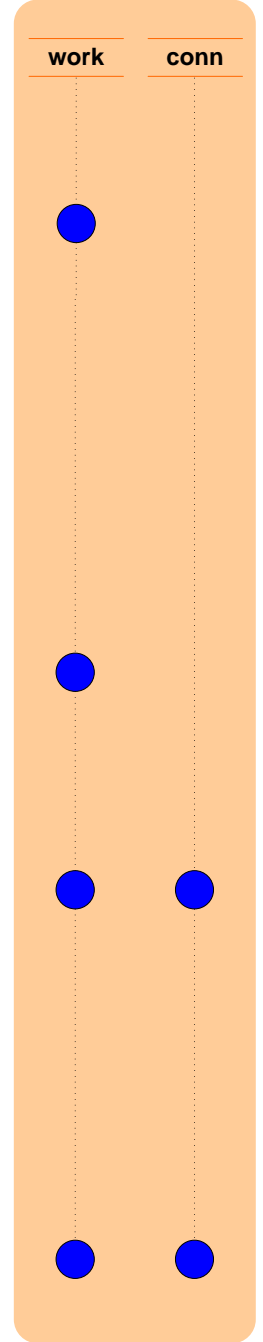
Directory Integrator

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.

Continue from last *Flow* component

Server Response
Hook Flow diagrams

Available temporary script variables

work	conn

Before Execute

Override Reply Enabled?

Yes

No

Output Attribute Map

Override Reply

Before Reply

Reply

After Reply

Reply Successful

Continue to start of next cycle

On Error

End-Of-Flow

IBM.
Directory Integrator

Continue from previous component or Start of AL

Available temporary script variables

work	conn	current
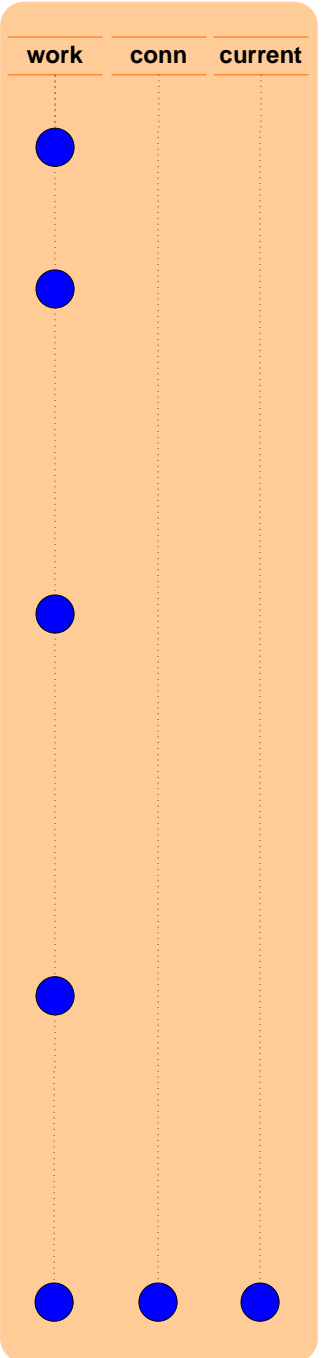
Available Objects

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 (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.).

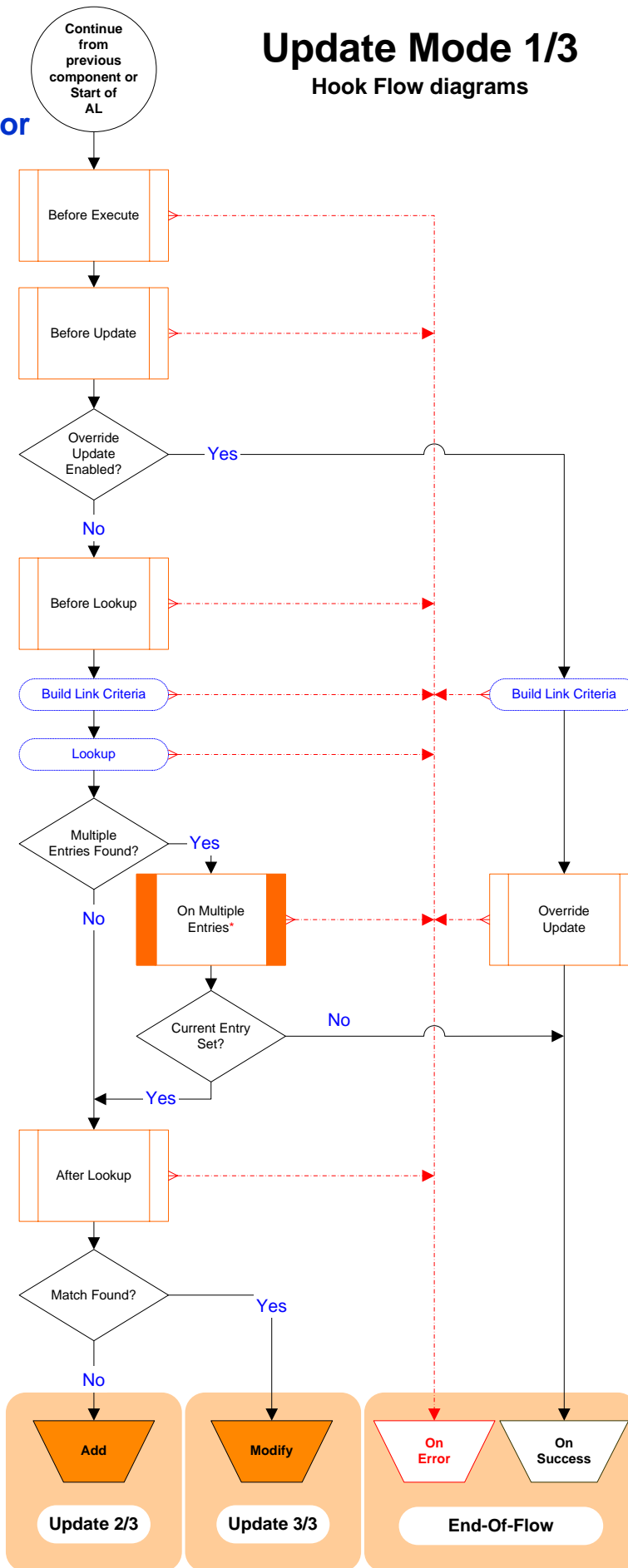In addition, **conn** may be set to the desired **Entry** object by calling the Connector's **setCurrent()** function:

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.

Flow:

Before Execute

Before Update

Override Update Enabled? — Yes / No

Before Lookup

Build Link Criteria

Lookup

Multiple Entries Found? — Yes / No

On Multiple Entries*

Override Update

Current Entry Set? — No / Yes

After Lookup

Match Found? — Yes / No

Add — Update 2/3

Modify — Update 3/3

On Error / **On Success** — End-Of-Flow

Hook Flow rev. 6.1
20060519

IBM.
Directory Integrator

Available temporary script variables

work   **conn**

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.

Override Add Enabled?

Yes

No

Add Override Script

Output Attribute Map

Before Add

conn Entry empty?

Yes

No

On No Add

Add

After Add

Continue to next component, next AL section, or start of next cycle

On Error

End-Of-Flow

After Update
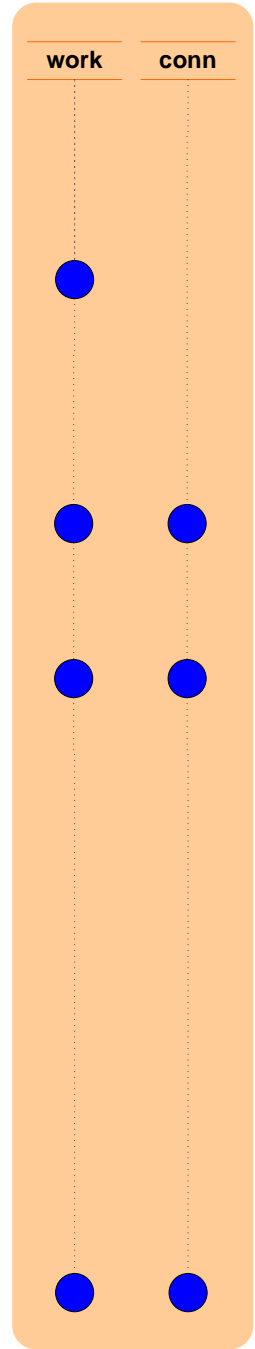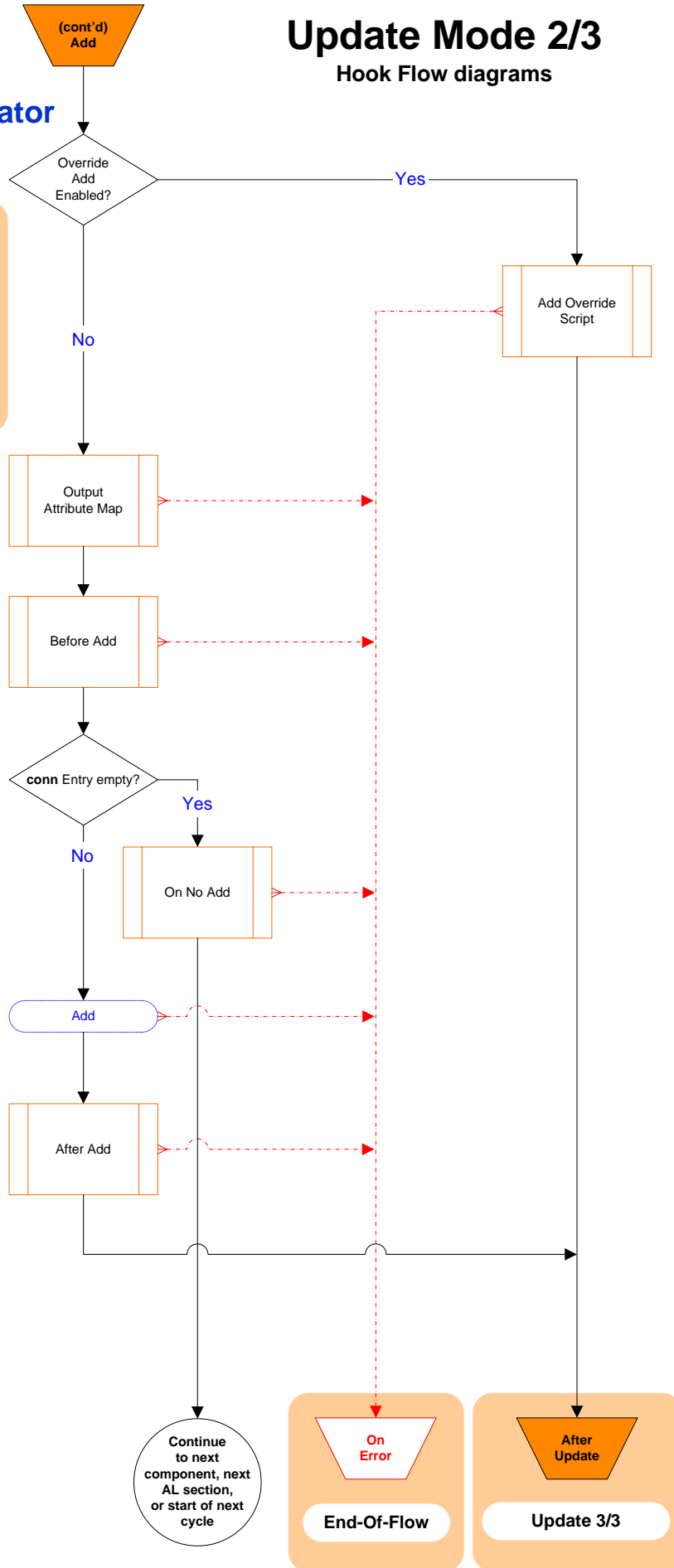
Update 3/3

Update Mode 3/3
Hook Flow diagrams

(cont'd) Modify

Available temporary script variables

work	conn	current

Override Modify Enabled? — **Yes** → Modify Override Script

No

Output Attribute Map

Before Modify

conn Entry empty?

Compute Changes?

Yes → Changes Detected? → **No**

No → On No Changes

Yes → Before Applying Changes

Modify*

(cont'd) After Update

After Modify

After Update

On Success

On Error

End-Of-Flow

Available Objects

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

If the update results in a **Modify** operation, the **current** object gives you access to the record/entry in the connected data source that matched the Link Criteria (e.g. is about to be modified). Note that until the Output Map, both conn and current contain the same information.

As in the case of an **Add**, the **conn** object holds the information that is to be written to the data source, in this case, by the **Modify** operation.
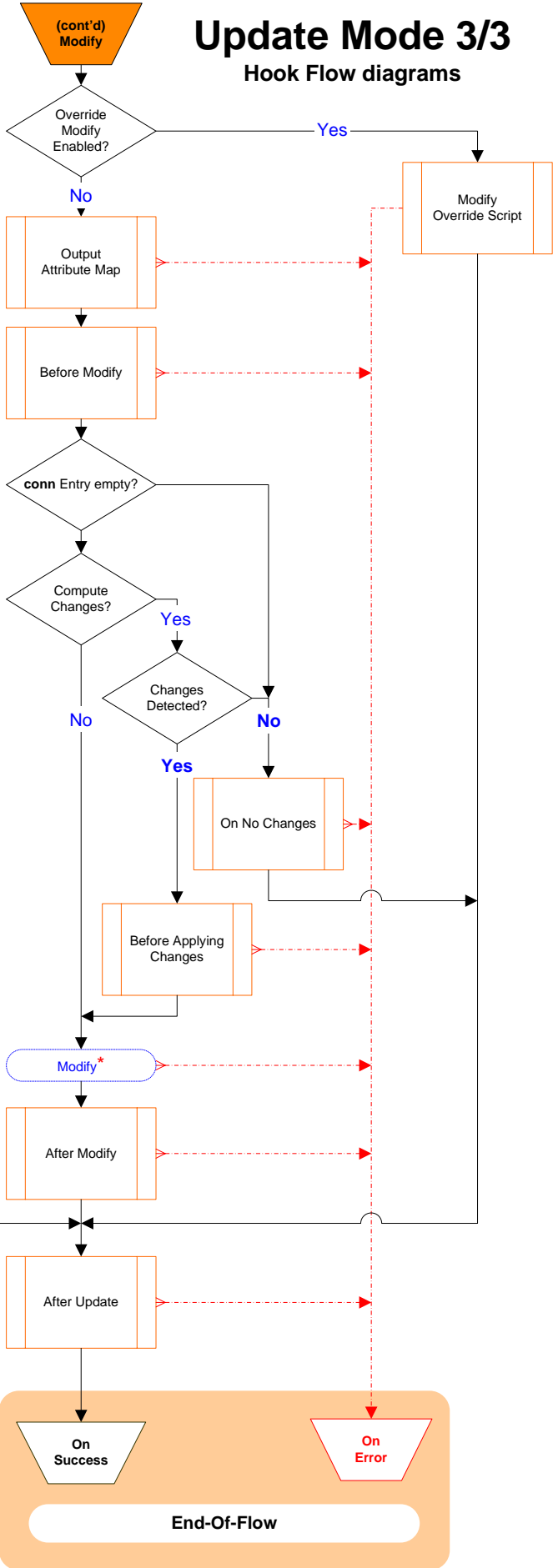
The *conn* object

The conn object is emptied immediately before the **Output Map**. After this point, **conn** and **current** no longer contain the Entry object found by the lookup operation.

*Modify

Please note that some data sources will compute changes automatically, and if none are detected, will revert with a **No Changes** exception. This will cause flow to be directed to the **On No Changes** Hook.
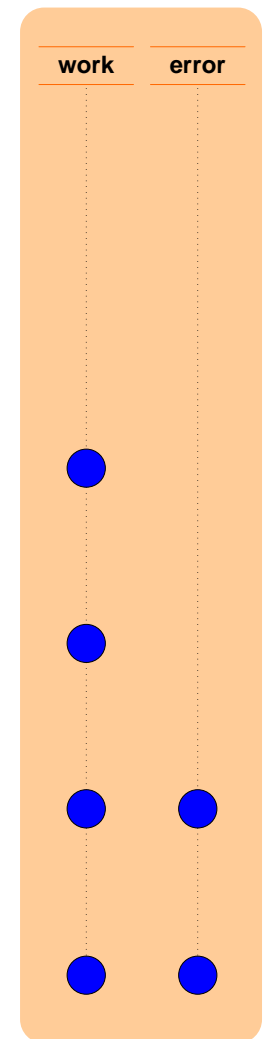
IBM.
Directory Integrator

End-Of-Flow for All Connector Modes
Hook Flow diagrams

Available temporary script variables

work   error

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 components that either terminate normally (e.g. successfully) or due to an error.

Error Handling

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

The error condition can be passed on to next On Error Hook (either the Default for the Connector, or the AssemblyLine Error Hook) by re-throwing the exception:

throw error.getObject("exception");

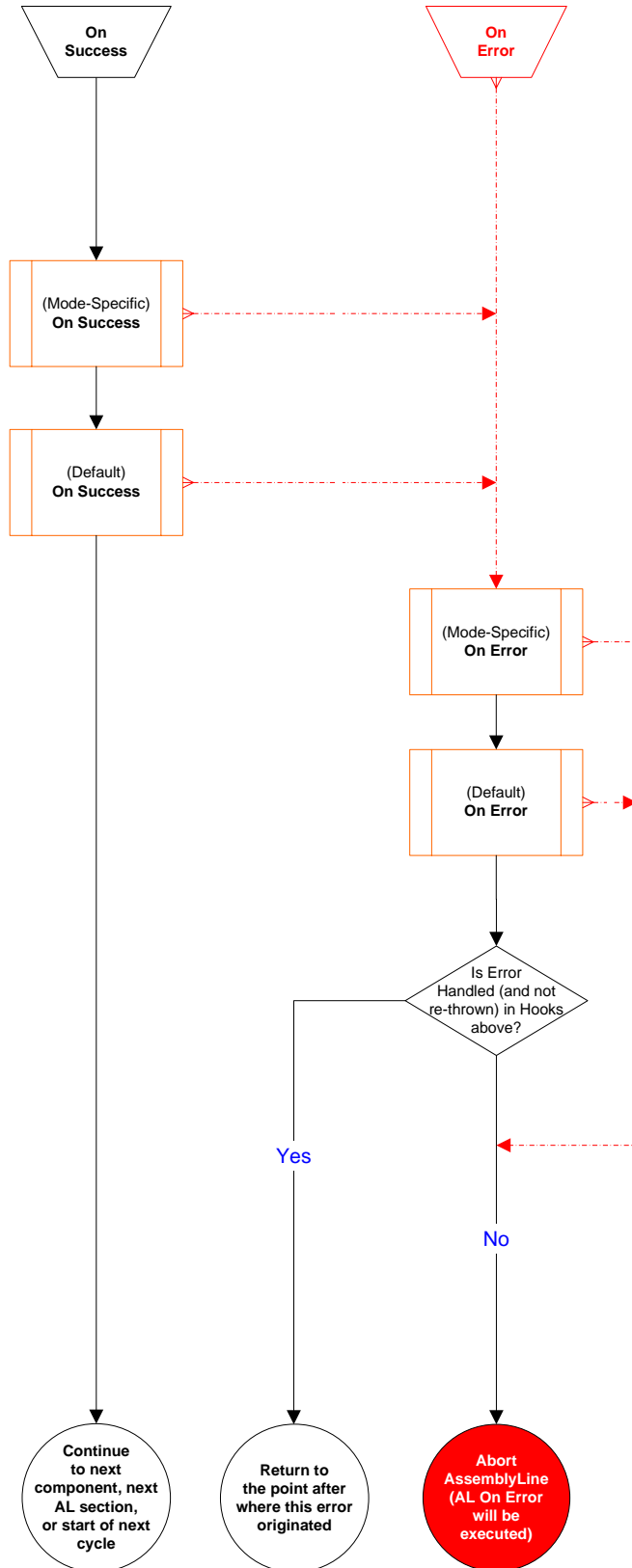Furthermore, if an error occurs in an **On Error** Hook, then the AssemblyLine will also **abort**.

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

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.

On Success

On Error

(Mode-Specific) **On Success**

(Default) **On Success**

(Mode-Specific) **On Error**

(Default) **On Error**

Is Error Handled (and not re-thrown) in Hooks above?

Yes

No

Continue to next component, next AL section, or start of next cycle

Return to the point after where this error originated

Abort AssemblyLine (AL On Error will be executed)

Hook Flow rev. 6.1
20060519

Connector Reconnect
Hook Flow diagrams

IBM
Directory Integrator

Available Objects

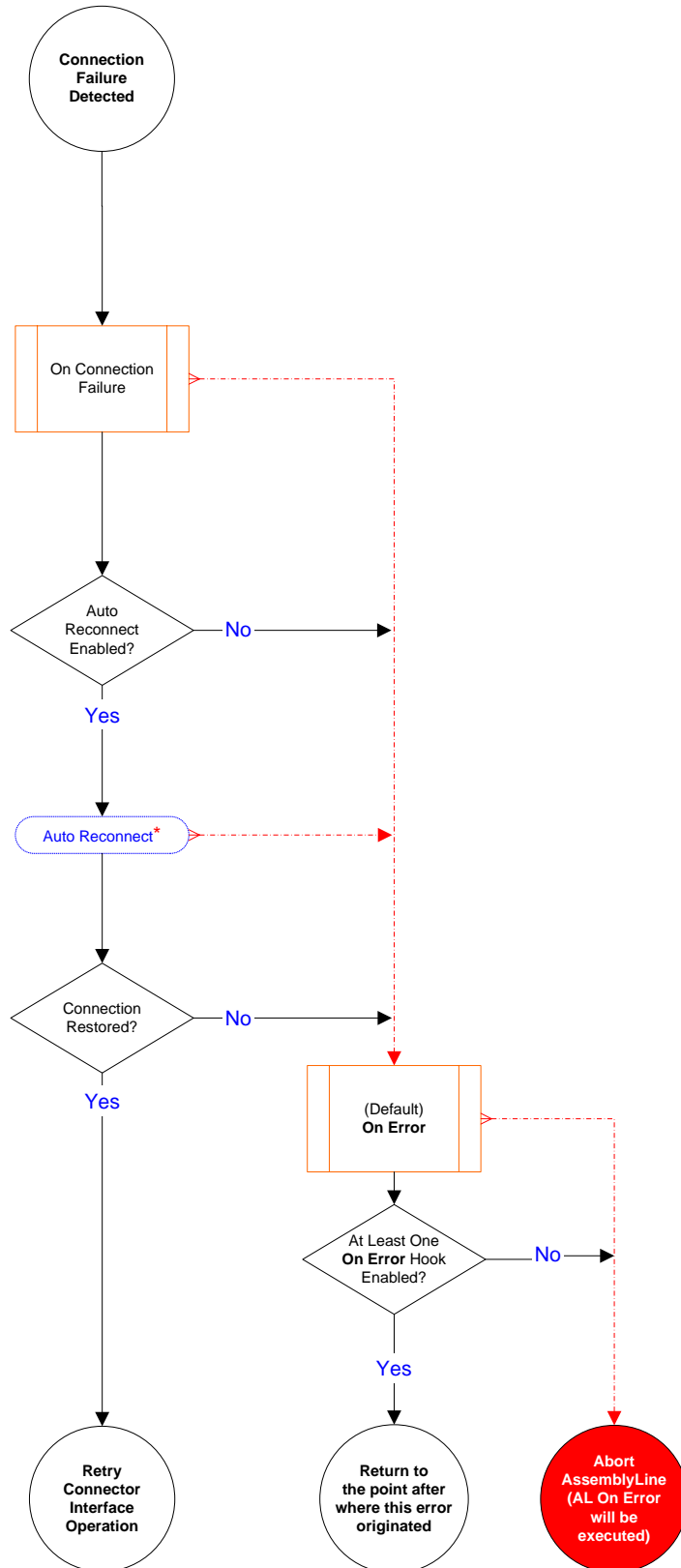As always, **work** gives you access to the attributes that are currently in the AssemblyLine,.

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 *connectorname*.

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.

Connection Failure Detected

On Connection Failure

Auto Reconnect Enabled? — No

Yes

Auto Reconnect*

Connection Restored? — No

Yes

(Default) **On Error**

At Least One **On Error** Hook Enabled? — No

Yes

Retry Connector Interface Operation

Return to the point after where this error originated

Abort AssemblyLine (AL On Error will be executed)

IBM.
Directory Integrator

Function (FC)

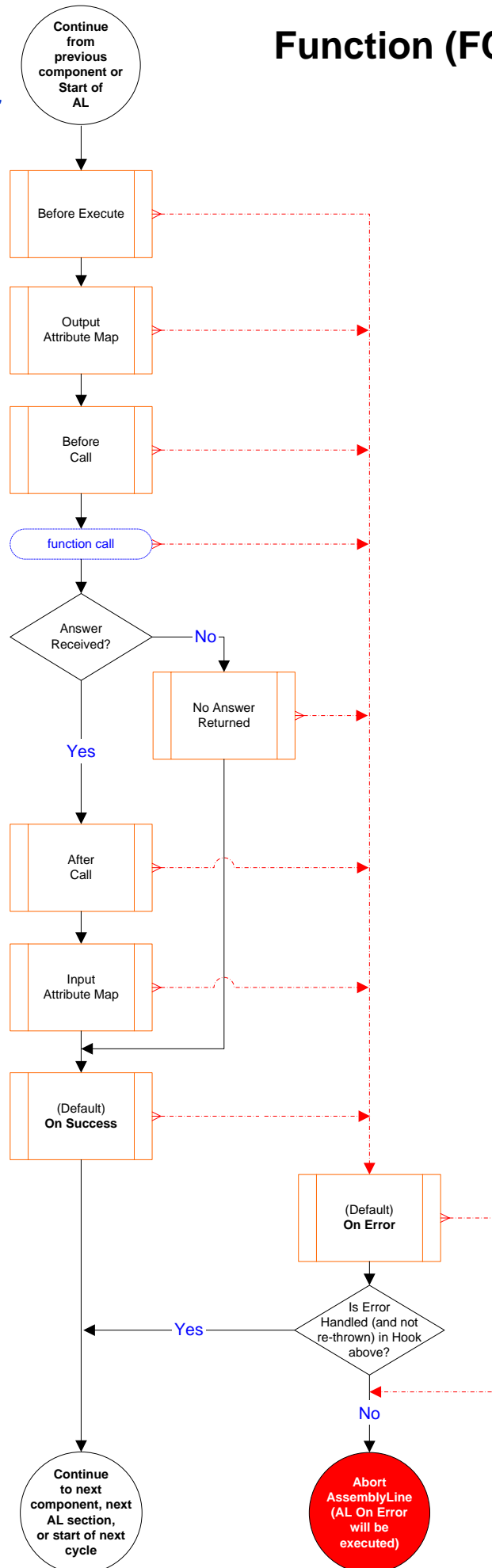**Continue from previous component or Start of AL**

Available Objects

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

*The information stored in the **conn** object changes during FC operation.

It is important to note that the **conn** object serves two different purposes in a **Function**:

1) Storing the call attributes/parameters defined in the **Output Attribute Map** to be transmitted by the Function call operation,

2) Receiving return attributes/parameters that will be mapped in by the **Input Attribute Map** after the Function call operation

Before Execute

Output Attribute Map

Before Call

function call

Answer Received?

No Answer Returned

No

Yes

After Call

Input Attribute Map

(Default) **On Success**

(Default) **On Error**

Is Error Handled (and not re-thrown) in Hook above?

Yes

No

Continue to next component, next AL section, or start of next cycle

Abort AssemblyLine (AL On Error will be executed)

Available temporary script variables

work	conn	error

*

Hook Flow rev. 6.1
20060519