Portrait
Foundation

Node Components Technical Reference

Edition 25.0

11 January 2013

Pitney Bowes
Software

Portrait Foundation
Node Components Technical Reference

Acknowledgement of trademarks

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK	**America**	**Norway**
Portrait Software	Portrait Software	Portrait Software
The Smith Centre	125 Summer Street	Portrait Million Handshakes AS
The Fairmile	16th Floor	Maridalsveien. 87
Henley-on-Thames	Boston, MA 02110	0461 Oslo
Oxfordshire, RG9 6AB, UK	USA	Norway
Email: support@portraitsoftware.com	Email: support@portraitsoftware.com	Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778	Tel: +1 617 457 5200	Tel: +47 22 38 91 00
Fax: +44 (0)1491 416601	Fax: +1 617 457 5299	Fax: +47 23 40 94 99

About this document

Purpose of document

This document describes the Portrait Foundation node components.

Intended audience

Anyone writing or using node components.

Related documents

EDGE2020 Implementation Guide *(EDGE-Portrait Implementation Guide)*

Software release

Portrait Foundation 5.0 or later.

Contents

1 Introduction

This document describes the Node Components that are available in the Portrait Foundation Configuration Suite Process Modeller. They are defined in different Portrait Platform packages which make up the structure of the Configuration Tree. Within the Configuration Suite, nodes appear in:

- **Supporting definitions—Palettes**, where the node is placed within a palette for use in the Process Modelling tool. The way the nodes are grouped within the palettes is described in section 2.
- **Supporting definitions—Models—Model node types**, where the node is defined. The way the nodes are grouped within the model node types is described in section 3.

A number of the nodes in the palette reference the following areas in the Configuration Tree:

- Authentication interactions; defined in **User interface—Authentication interactions**.
- Custom interactions; defined in **User interface—Custom interactions**.
- Custom nodes; defined in **System extensions—Custom nodes**.
- Code node plug-ins; defined in **System extensions—Code node plug-ins**.
- Generated interactions; defined in **Generated interactions—Domains**.
- Scripts; defined in **Supporting definitions—Scripts**.

1.1 Assumptions

1. The **Verify()** method on nodes is used to verify the internal structure of a node, that is the necessary member variables have been initialised and are valid.
2. The **Validate()** method on nodes is used to determine whether or not the use of the node breaks the model.

2 Palette groupings

The nodes components are grouped as follows in the palette:

Palette	Node
Process flow	• Start • End • Branch • Branch conditionally • Supersede • Manipulate model data • Data object classification • Get data object from collection • Add data object to collection • Delay • Or • And • Loop start • Loop end
Portrait nodes	• Run model • Run model via class • Run external code defined in **System extensions—Custom nodes** • Placeholder • Access globals • Get smart lookup value 2 • Run scripts defined in **Supporting definitions—Scripts**
Data access nodes	• Access • Access data via class • Save composite data object • Retrieve composite data object
Security nodes	• Deny model execution • Permit model execution
Custom nodes	• Custom nodes defined in **System extensions—Custom nodes**
User interface nodes	• Display generated interaction • Display custom interaction • Display custom interaction via class • Display authentication interaction • Generate menu • Retrieve GI answers • Retrieve GI view details • Format view of CBE

Palette	Node
Event processing nodes	• Create change notification subscription • Publish change notifications • Consume change notifications • Create external event • Trigger external event • Wait for external event • Reach milestone • Retrieve client event subscription
Workflow nodes	• Put back task • Create task • Update task • Resume task • Suspend task • Get task • Save task • Complete task • Create auto task
Document management nodes	• Create outbound document • Send all from outbox • Remove all from outbox • Retrieve document • Save document • Search for documents • Retrieve envelope • Save envelope
EDGE integration nodes	• EDGE Logon • EDGE Logoff • EDGE Guide • EDGE Logic • EDGE Project list • EDGE Autoreceive • EDGE Queue list

3 Model node type groupings

The nodes components are grouped as follows in **Supporting definitions—Models—Model node types**.

Group	Node
Case	• Reach milestone
Client events	• Retrieve client event subscription • Send client event
Custom	• Custom nodes defined in **System extensions—Custom nodes**
Document management	• Associate document • Export document content • Remove document associations • Retrieve document • Retrieve envelope • Save envelope • Search for documents
EDGE integration nodes	• EDGE Logon • EDGE Logoff • EDGE Guide • EDGE Logic • EDGE Project list • EDGE Autoreceive • EDGE Queue list
External events	• Create external event • Trigger external event • Wait for external event
Portrait	• Access globals • Add data object to collection • Data object classification • Get smart look-up value (deprecated but included for backward compatibility) • Get data object from collection • Get smart lookup value 2 • Manipulate model data • Placeholder • Run external code • Run model • Run model via class • Supersede

Group	Node
Process flow	• And • Branch • Branch conditionally • Delay • End • Loop end • Or • Start • Loop start
Scripting	• Scripts defined in **Supporting definitions—Scripts**
Security	• Deny model execution • Permit model execution
Simplex channel	• Create outbound document • Remove all from outbox • Send all from outbox
User interface	• Consume change notifications • Create change notification subscription • Display authentication interaction • Display custom interaction • Display custom interaction via class • Display generated interaction • Format view of CBE • Generate menu • Publish change notifications • Retrieve GI answers • Retrieve GI view details
Workflow nodes	• Complete task • Create auto task • Create task • Get task • Put back task • Resume task • Save task • Suspend task • Update task

4 Portrait node components

4.1 Add data object to collection

Model node type group	Portrait		
Palette	Data nodes		
Description	This node is used to add a single data object element into a data object collection. The data object collection will either be provided or created for the user of this node. (See node Inputs section.)		
Label	This should be taken from the node configuration. If no configuration is provided, this should default to **Add Data Object To Collection**.		
Verification Rules	None.		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration(not currently Implemented)			
Label	This is the text that should be shown in the label.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
Collection Element	Same type as the collection elements	Yes	This is the element to be added to the collection.
Collection	Collection (of data objects)	No	This is the collection that is to have the collection element added to it. If it is not supplied, a new collection will be created by the node and the collection element will be added to the new collection.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Collection	Collection (of data objects)	N/A	This is the collection that has had the collection element added to it.
Outcome outputs	**Fail**		
None			

4.2 And

Model node type group	Control nodes
Palette	Control nodes
Description	This node is used to manage the combination of dependency arcs. The node waits until all the connected nodes complete with the specified outcomes.
Label	NLS text for **And.**
Validation Rules	Each and node must have >=1 Arc entering and 1 arc leaving.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A

None

4.3 Associate document

Model node type group	Document management
Palette	Document management nodes
Description	This node creates the associations that link a document to other entities. The following many-to-many associations are supported. • Party • Contract • Case [future functionality] An association is created by passing in a collection of Ids for each relevant entity. The node adds an association to the entity referenced by the ID. If an ID is passed in for a duplicate association, the node still completes successfully. The duplicate reference is ignored. If an invalid ID is passed in, the node will still create the other valid associations. A Fail outcome is returned, together with a collection containing all invalid references.
Label	Associate document
Verification Rules	There must be at least one entity to be associated, so at least one of Parties or Contracts must be mapped in the inputs.
Validation Rules	The node must have one arc entering and two arcs leaving.

Node configuration

None

Inputs

Name	**Type**	**Mandatory**	**Description**
Document ID	String	Yes	Unique identifier of the document being associated.
Parties	Collection	No	A collection of Party Reference data objects for the associations that are to be created. The Party Reference data object properties are as below. Party ID String. This is mandatory
Contracts	Collection	No	A collection of Contract Reference data objects for the associations that are to be created. The Contract Reference data object properties are as below. Contract ID String. This is mandatory

Outcome outputs **OK**

Name	**Type**	**Mandatory**	**Description**
Document ID	String		The system returns the Document ID if all References have been added successfully.

Outcome outputs **Fail**

Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible Fail codes are as follows. Invalid reference. One or more of the references passed in is invalid. The Failed References are returned in the Failed References collections Empty inputs. All the collections passed in were empty. No associations were created. Unknown error. The transaction has failed for an unknown reason. No associations have been created. No Failed Reference collection is returned.

Invalid references	Data object collection	If any of the references are invalid, the node will return an Invalid References collection. The Invalid Reference data object has the following properties. Object type (i.e. one of Party or Contract) Object ID

4.4 Display authentication interaction

Model node type group	Portrait
Palette	Authentication interaction nodes
Description	This node is used to present an authentication interaction UI in a channel. The node prepares user interface information for the channel enabler. The channel enabler then renders a user interface while the node waits for the client to return. Once the client returns, the node then compares the returned data with data retrieved for the identified Configurable Business Entity (CBE). The entity can be identified before the user interface is constructed (node input) or when the client returns (data entry). The latter of these two options can only be used if none of the comparison require entity data before the data entry display is built (e.g. Two letters from password). The node invokes display and comparison methods on each comparison strategy component that is referred to in the questions configured into the authentication interaction.
Label	The NLS specific name of the selected interaction definition.
Node configuration	
Authentication interaction	This is a combo box built by listing all the authentication interactions under the User interface section of the current PWS file. The list shows the latest versions of all published interactions, as well as the version already referred to by the node (if any). The list will also show any draft versions in the current snapshot.
Do not display	This is a check box. When set, all the optional inputs on the node become mandatory. The flag is disabled if any of the questions have the Must be asked characteristic set to True.

Inputs (variable)

Name	**Type**	**Mandatory**	**Description**
ID	Varies. This is taken from the data type of the selected identification attribute.	Yes (if present)	This is any valid unique attribute for the party. This input is based on the Identification attribute characteristic configured in the interaction definition. The input is shown on the node if none of the configured questions refer to it or if any of the questions refer to a comparison strategy that requires the ID up front. If present, the node retrieves data for each of the comparison attributes for the specified party. The node uses the ID supplied + the Party Type configured to retrieve the data. If this fails, then the node fails.
Questions - a number of inputs.	Varies. This is taken from the data type of the question	No, unless Do not display flag is set.	There will be an input for every question configured in the AIN, unless the Must be asked characteristic on the question is set to true. If a value is passed into the node, then the strategy will not display the question in the UI. Otherwise it will. This decision is based on Value rather than the Mapping. If the input is mapped, but the context item is empty, then the question must be presented in the UI.

Outcome outputs **OK**

Name	**Type**	**Mandatory**	**Description**
Entity ID	String	Yes	The ID of the authenticated entity.

Entity Category			This is the entity’s category as configured in the AI definition.
Entity Type	String	Yes	This is the configured entity type as configured in the AI definition.
Outcome outputs	**Fail**		
None			

4.5 Branch

Model node type group	Control nodes
Palette	Control nodes
Description	This node branches a single arc into multiple arcs in order to satisfy the dependency of many nodes on a single outcome.
Label	NLS text for **Branch**.
Verification Rules	None.
Validation Rules	Only 1 arc entering, at least 2 arcs leaving.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A
None	

4.6 Retrieve client event subscription

Model node type group	General nodes
Palette	Portrait nodes
Description	This node is used to retrieve the number of subscriptions to a specified client event type and optionally a particular client event identifier..
Label	Client Event Subscription Information
Verification Rules	None.
Validation Rules	1 Each node must have 1 arc entering and 1 or 2 arcs leaving.
Node configuration	
None	

Inputs			
Name	**Type**	**Mandatory**	**Description**
EventType	String	Yes	The event type is the system name of the reference data item in the client events reference data group.
EventID	String	No	This is the qualifying data for the event Type. Typically, this would refine the Event Resolution process. For example, if the Event Type was “Consumer”, the event ID may well be the Consumer ID.
Outcome outputs	Subscription information		
Count	Integer	-	The number of subscriptions currently active.
Outcome outputs	No subscriptions		
None			

4.7 Complete task

Model node type group	Workflow nodes
Palette	Workflow nodes
Description	The **Complete task** node is used to complete a CTN. The node takes a Task ID as its input and completes it using the outcome configured in the node's configuration.
Label	**Complete task**.
Validation	The node's configuration reads available outcomes from a selected task type. If the task identified by the Task ID is not the same type as the configured task type, then the node will fail. If the **TaskInstance** is not checked out to the party performing the operation, the node will not complete the task. If status of the task was not **active**, OR **passive** OR **suspended**, the node will not complete the task and will reach **NoAccess** outcome instead.
Node configuration	
Task definition	The list of task definitions is provided to select desired task definition matching the task being completed.
Outcome	The list of outcomes taken from the task definition configuration is provided to select the desired outcome **Task Instance** (for the resumed **Create task** node).

Inputs

Name	**Type**	**Mandatory**	**Description**
TaskInstance	DataObject	Yes	Contains information about the task being completed.
PartyID	String	Yes	Party performing CompleteTask operation.
PartyType	String	Yes	Party performing CompleteTask operation.
Description	String	No	Notes that should be added to the task history.

Outcome outputs	**OK**
None	
Outcome outputs	**NoAccess**
None	
Outcome outputs	**Fail**
None	

4.8 Branch conditionally

Model node type group	Control nodes
Palette	Control nodes
Description	This node is used to branch models when specified conditions apply
Label	This should be taken from name property in the node configuration.
Node configuration	
Name	This is the text that should be shown in the label.
Description	This is used to hold a general description of the purpose of the branching. This text must be available for use in generated documentation.
Outcomes	The list of outcomes applied by the node. Each outcome consists of a reference to an outcome definition and a condition. The user can create new outcomes, edit existing outcomes and delete them. The user can also create new conditions through the condition editor.
Inputs	
Inputs are derived from configuration	

Outcome outputs	Outcomes are derived from configuration		
Name	**Type**	**Mandatory**	**Description**
None			

Outcome outputs	Default		
Name	**Type**	**Mandatory**	**Description**
None			

Outcome outputs	Fail		
Name	**Type**	**Mandatory**	**Description**
Failed condition	**String**	**n/a**	**The name of the condition that caused the failure.**

4.9 Consume change notifications

Model node type group	General
Palette	Portrait
Description	The node returns a collection of change records from those present in the session context, if any. After being read from the context, the records are deleted. If no change records are present in the session context, the node returns the **NoChange** outcome and an empty change record collection. If change records are present the OK outcome is returned along with a collection of the change records.
Label	**Consume change notifications**
Verification Rules	None
Validation Rules	The node must have one arc entering and two arcs leaving.

Node configuration
None

Inputs
None

Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
ChangeItems	Collection		A collection of change records.

Outcome outputs	**No change**		
Name	**Type**	**Mandatory**	**Description**
ChangeItems	Collection		An empty collection

4.10 Create auto task

Model node type group	Workflow nodes
Palette	Workflow nodes

Description	The CATN is really a submodel node with a difference. The node refers directly to a model, which will be used to execute the task automatically. The models referred to are Business Operation Models (BOMs) and these must never contain any User Interface Nodes, since there will be no client attached when the model is run. Since the node refers directly to a model, the node's Inputs, Outcomes and Outputs are derived directly from the model and the node can be mapped in exactly the same way as a synchronous submodel node. When the node fires, a task is created in the normal way in the database. The input context is stored as a task attribute and is passed onto the submodel when the task becomes active. When the task becomes active it will be detected by a service which will then run the associated model. The model will be passed the input context, which was saved with the task and, when the model completes, the service will complete the CATN with the appropriate outcomes and outputs		
Label	Locale specific version of **CATN**.		
Verification Rules	A submodel must be selected for the node. The Originator Party ID and Originator Party Type are supplied at runtime, along with some other fixed inputs that relate to creation of the task. Also mandatory inputs to the submodel have to be supplied. The inputs of the submodel must not conflict with the fixed inputs defined below for the node.		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
Delay for	Specifies a delay interval after that the task is eligible for completion.		
Execution Model	Model triggered at the time of completion of the task.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
Originator Party ID	String	Yes	Identifier of the user performing creating the task.
Originator Party Type	String	Yes	This is the system name of the party that Party ID is an instance of.
ActivationDelay	Integer	No	This specifies the number minutes to delay before activating the task, i.e. before the model is eligible to run. If this is passed it overrides the configured **Delay for** interval.
Associations	Data Object Collection	No	Collection of RepeatingAttributes to associate with the created task instance id. Each RA passed is updated and saved.
ExternalEventID	String	No	If an External Event Id is passed here, then when the task has been created the External Event will be triggered. Passed as trigger data to the External Event will be a Data Object of category Workflow and type CreateTaskTriggerData, containing the task instance id of the new task
AttributeInfo	Data Object	No	The data object that describes the selected task definition. Category='TaskDefinition', Type = <TaskTypeName>.
Outcome outputs	As of the referenced submodel		
Name	**Type**	**Mandatory**	**Description**
As of the referenced submodel	As of the referenced submodel	As of the referenced submodel	As of the referenced submodel.
Outcome outputs	**Fail**		
None			

4.11 Create change notification subscription

Model node type group	General

Palette	Portrait		
Description	The node returns a set of change records defined by the set of configured change categories. Conceptually it is similar in operation to the publisher node immediately followed by the consumer node, except that the change records are not written to the session context. The node always returns the OK outcome.		
Label	**Create change notification subscription**		
Verification Rules	None.		
Validation Rules	The node must have one arc entering and one arc leaving.		
Node configuration			
Selected change categories			
Inputs			
Variable, depending on the selected change categories and the parameters declared thereon.			
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
ChangeItems	Collection		A collection of change record data objects.

4.12 Create external event

Model node type group	General		
Palette	External event nodes		
Description	This node component is responsible for creating a new external event instance in the database with a unique identifier. The initial state will be **Created** and a new history record will be created showing the creation of this external event. Two mandatory values must be provided at creation time, the event type and event subtype. A check is made against the **ExternalEventType** and **ExternalEventSubtype** reference data groups to ensure they are valid values. Optionally a contract identifier or party information can be provided and associated with the new event. Note that for party information the identifier and type must either both be present or both absent.		
Label	NLS text for **Create external event**.		
Verification Rules	None.		
Validation Rules	List break Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Event type	String	Yes	The identifier (RDI) of the event type (for example Receipt of document).
Event sub type	String	Yes	The identifier (RDI) of the event subtype. This provides more information about the type of event (for example driver's licence) and may be used to match an external event if the event identifier isn't available.
Party ID	String	No	The identifier of the party to which the event relates. This identifier will be used to help match an external event to a party if the event identifier isn't available.
Party Type	String	No	The type of party to which the event relates.
Contract ID	String	No	The identifier of the contract to which the event is related.
Outcome outputs	**OK**		

Name	Type	Mandatory	Description
External Event ID	String	N/A	The identifier of the created event record.
Outcome outputs	**Fail**		
None			

4.13 Create outbound document

Model node type group	Simplex channel
Palette	Simplex
Description	This node component is used to place a document in the simplex channel outbox.
Label	Configured document name.
Verification Rules	1 A document (as listed in the Documents section of the project) must be specified. 2 The document must exist in the Configuration Tree. 3 A Simplex Channel must be specified. 4 The Simplex Channel must exist in the Configuration Tree.
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
None	
Inputs	

Name	Type	Mandatory	Description
Engagement Session ID	String	Yes	This is the identifier of the current engagement session. The **Start Engagement Session** DAT will return this value.
Language	String	Yes	This is the language of the document.
Recipient Party ID	String	Yes	This is the identifier of the party selected to receive the document.
Recipient Party Type	String	Yes	The recipient party's type.
Document data	Data object	Yes	This data object contains properties for each of the document's fields. The contents of this data object vary according to the document selected.
Envelope data	Data object	Yes	This document contains properties for each of the fields associated with the selected channel. Consequently, as above, the data object varies according to the channel selected.
Channel control	Data object	No	This data object contains properties used by the associated media producer. In the case of document printing for the post channel, the channel control object has to contain a property called **Location** specifying the logical name of configured printer. The printer location is retrieved by Printing Media Producer and the desired physical printer is asked to print the document.
Covering paragraph name	String	No	This parameter identifies the covering text that should be inserted into the selected channel's covering letter. The actual text inserted will be held as reference data in the Cover Text Reference Data Group. The reference data group will contain a language key so that the selected text will vary with language.
Send Immediate	Boolean	No	This parameter identifies whether to flag the document to be sent immediately. If not supplied will default to FALSE.
Contact History Text	String	Yes	This is the text required for setting up contact history for the document.

Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Document ID	String	N/A	This identifier uniquely identifies the document instance and must be available as a value to be used in the document template. The identifier will typically be passed to a **Wait for external event** node, which will then write the identifier into the event lookup table. For the document receipt event to be registered successfully, the Document ID will need to be read from an inbound document (for example from a bar-code).

Outcome outputs	**Failed**
None	

4.14 Create task

Model node type group	Workflow nodes
Palette	Workflow nodes
Description	This node component is used to create a task in the Task Engine. Once a task is created, the node in the model will not complete until the task is executed or superseded. In order to execute the task, the task will either be picked through the Pick Work dialog, or retrieved by a call to Get Task. The task definition has a number of properties with configured values. Each of the values can be overridden in the node configuration.
Label	Locale sensitive version of: **Create task -** plus the name of the selected task definition.
Verification Rules	1 A Create Task Node must be associated with a task Definition. 2 The Task Definition must exist in the Configuration Tree.
Validation Rules	Each code node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
Task definition	This identifies the type of task to be created. A list of available tasks is provided from the workflow/task definition section of the **project.amc** file.
Initial priority	Overrides the corresponding task definition property. The value options should be presented using the metadata for the Task Definition Node Type.
Escalation value	Overrides the corresponding task definition property. The value options should be presented using the metadata for the Task Definition Node Type.
Escalation interval	Overrides the corresponding task definition property. The value options should be presented using the metadata for the Task Definition Node Type.
Reroute limit	Overrides the corresponding task definition property. The value options should be presented using the metadata for the Task Definition Node Type.
Activation delay	Overrides the corresponding task definition property. The value options should be presented using the metadata for the Task Definition Node Type.

Inputs			
Name	**Type**	**Mandatory**	**Description**
OriginatorPartyID	String	Yes	The Id of the party that created the task.
OriginatorPartyType	String	Yes	The party type that created the task.
Initial Status	String	No	The initial status for the task (see also **Outcome**)
SystemNameOfPartyType	String	No	The party type that the task should be routed to if RoutedToPartyId is specified.

RoutedToPartyId	String	No	The party Id of the party to route the task to.
AdditionalRoutedToParties	Data Object Collection	No	An additional collection of routed to parties. Objects in the collection must contain attributes 'PartyID' and 'PartyType'
Priority	Integer	No	Used to override the value defined in the node configuration. (See above)
ActivationDelay	Integer	No	Used to override the value defined in the node configuration. (See above)
EscalationValue	Integer	No	Used to override the value defined in the node configuration. (See above)
EscalationInterval	Integer	No	Used to override the value defined in the node configuration. (See above)
RerouteLimit	Integer	No	Used to override the value defined in the node configuration. (See above)
Execution Windows	Data Object Collection	No	Specifies execution windows for the task.
Deadline	Data Object	No	Specifies deadline information.
Associations	Data Object Collection	No	Collection of RepeatingAttributes to associate with the created task instance id. Each RA has is updated and saved.
Outcome	String	No	If the **Initial Status** input is set to COMPLETED, this defines the outcome that will be set for the task.
ExternalEventID	String	No	If an External Event Id is passed here, then when the task has been created the External Event will be triggered. Passed as trigger data to the External Event will be a Data Object of category Workflow and type CreateTaskTriggerData, containing the task instance id of the new task
AttributeInfo	Data Object	Yes	The data object that describes the selected task definition. Category='TaskDefinition', Type = <TaskTypeName>
Outcome outputs			
Name	**Type**	**Mandatory**	**Description**
Outputs	As defined in the task definition	As defined in the task definition	The outputs to this node are the outputs of the task definition, encapsulated in the Attribute Info DO
Outcome outputs	**Fail**		
None			

4.15 Display custom interaction

Model node type group	Portrait
Palette	Custom interaction nodes
Description	This node is used to insert a user interface point into a model. The node points to a custom interaction, which defines the node's inputs, outcomes and outputs. When the node fires, the channel infrastructure retrieves a URL (from a custom interaction Screen definition) using the identifier of the definition, the current channel and the current language and redirects the browser to that page. When the page is submitted the browser runs the Resume Model page, passing an outcome and a set of outputs, which should conform to the custom interaction (this cannot be guaranteed). The resume model page resumes the model by setting the custom interaction node's outcome and outputs accordingly.
Label	The name of the configured custom interaction.
Verification Rules	1 A custom interaction must be associated with the Node. 2 The Custom interaction must exists in the Configuration Tree.

Validation Rules	Each client action node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
Custom	This is the selected custom interaction. The selection determines the node's inputs, outcomes and outputs.
Inputs	
Determined by configuration	
Outcome outputs	Determined by configuration
Determined by configuration	

4.16 Display custom interaction via class

Model node type group	Portrait
Palette	Custom interaction nodes
Description	This node is used to insert a user interface point into a model. The node points to a custom interaction class, which defines the node's inputs, outcomes and outputs. When the node fires, the input parameter 'alias' is used to determine which custom interaction to trigger. Mappings are made between the class and the custom interaction's inputs according to the definition in configuration. The node then suspends and the normal custom interaction process if followed until the node resumes. At this point the outcome and outputs are mapped from the custom interaction to the class's outcome and outputs according to the definition in configuration. If an outcome cannot be resolved back to a class outcome or if any other error in mapping occurs the node resumes the model with an outcome of 'Class mapping failed'
Label	The name of the configured custom interaction.
Verification Rules	1 A custom interaction class must be associated with the Node. 2 The Custom interaction class must exists in the Configuration Tree.
Validation Rules	Each custom interaction class node must have 1 arc entering and >= 2 arcs leaving.
Node configuration	
Custom interaction class	This is the selected custom interaction class. The selection determines the node's inputs, outcomes and outputs.

Inputs			
Name	**Data type**	**Mandatory**	**Description**
Custom interaction alias	String	Yes	The system name of the custom interaction reference as defined for the custom interaction class.
Other inputs determined by configuration	-	-	-
Outcome outputs	Determined by configuration		
Class mapping failed			Outcome if an error in mapping occurs at runtime.
Determined by configuration			Other outcomes / outputs are configured for each custom interaction class.

4.17 Access data

Model node type group	Portrait
Palette	Portrait nodes

Description	This node is used to link a Data Access Transaction (DAT) into a model. The node reads the DAT definition, which determines the node's inputs, outcomes and outputs. DATs are documented separately.
Label	Name of configured DAT.
Rules	Each node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
System	Select Portrait or any external system available. The Transaction list below will contain all DAT(s) for the selected system.
Transaction	This is the identifier of the DAT. The DAT is selected from a list of DATs declared in the configuration tree. Once the DAT is specified, the node reads its input, outcome and output specification.
Inputs	
Determined by configuration	
Outcome outputs	Determined by configuration
Determined by configuration	

4.18 Access data via class

Model node type group	Portrait
Palette	Portrait nodes
Description	This node is used to execute some processing that is determined at runtime that supports a specific interface defined at configuration time. The processing that can be executed can either be a Data Access Transaction (DAT) or a Data Access Model (DAM). At configuration time, the data access class is configured with data access class inputs, outcomes, outputs and class members. Members of the data access class can be of two types; transaction members and model members. The list of candidates for model members is Data Access Models. The list of candidates for transaction members are all Data Access Transactions, At runtime the input 'Class Member' is used to select the member from the configured class. If this name is not a member then the 'Not A member' outcome is triggered. The inputs to the member are created by processing the mappings created at configuration time for the member. Once the member has completed the mappings created at configuration time are used to produce the outcomes/outputs for the class node from the outcome/outputs of the member. The data access class node always executes its members synchronously.
Supersede	If the member is a DAT no action occurs. If the member is a model the behaviour is the same as for a submodel node.
Label	The name and version of the model class node.
Verification Rules	A data access class must be associated with the node. The data access class must exist in the configuration tree.
Validation Rules	Same as for the submodel node.
Node configuration	
Model class	This is the selected data access class. This selection will contribute to the determination of the inputs, outcomes and outputs of the data access class nodes.
Description	Once the model class has been selected, its description is displayed in read-only form so that the configurer is informed of the function of the selected data access class.

Inputs

Name	Type	Mandatory	Description
Class member	String	Yes	System name of the data access class member that is defined

			for the data access class.
Other inputs			Determined by the configuration of the data access class (I.e. its interface)..
Outcome outputs	**Not a member**		
Name	**Type**	**Mandatory**	**Description**
Display name	**String**		**Display name**
System name	**String**		**System name**
Version number	**String**		**Version number**
Outcome outputs	Other		
Name	**Type**	**Mandatory**	**Description**
			The outcomes/outputs are as specified in the data access class definition

4.19 Manipulate model data

Model node type group	Data nodes
Palette	Data nodes
Description	This component allows the user to create new data items that are fully specified (that is if it is a Data object they will have a value for Category and Type) and allow items in the data context of a model or constant values to be used to populate them. The node allows any number of new items to be created and populated. The items are populated using the map data inputs dialog. They are mapped back into the context using the map data outputs dialog. Any data item created and not initialised with another item from the data context or a constant value will be assigned a value of null.
Label	**Data manipulation**.
Verification Rules	None.
Validation Rules	Each node must have 1 arc entering and 1 arc leaving.
Node configuration	
Output values	The outputs from the node are determined by the node configuration. When you edit this node you will be presented with the dialog shown below. This allows you to define a number of data items. These items will be the outputs from the node.
Inputs	
Input values	The inputs to the node are determined by the node configuration. When you edit this node you will be presented with the dialog shown below. This allows you to define a number of data items. These items will be the inputs to the node.
Outcome outputs	N/A

4.20 Data object classification

Model node type group	Data nodes
Palette	Data nodes

Description	This component is used to classify a data object fully or partially so that its exact Category and optionally Type can be determined. An unclassified or partly classified data object (directly or within a data object collection) is passed in as an input and the output is a fully or partly classified object. The node supports two types of comparison – **Is A** for a direct comparison of the data object category and type and **Is Derived From** for a comparison with hierarchy relations applied. If **Is Derived From** comparison is selected, the test succeeds if the data object is of the specified type or if its definition is a specialization of the specified type. The node can have configured more than one test rule. The rules are ordered by the priority. The first successful test, the one with the highest priority, determines the outcome of the node. The node will always provide a default outcome called **Other** if the object is not one of the types specified in the node configuration properties dialog (see below).
Label	**Data Object Classification**.
Verification Rules	None.
Validation Rules	This node must have 1 Arc entering and >=1 arcs leaving.

Node configuration

Output values	The output from this node is determined dynamically. It will be one of the data objects of the appropriate category and type that the user is interested in, or nothing.

Inputs for input type Data Object

Name	**Type**	**Mandatory**	
Unknown Data Object	Data Object	Yes	Data object to be classified

Inputs for input type Data Object Collection

Name	**Type**	**Mandatory**	
Unknown Collection	Data Object Collection	Yes	Collection, which contains the data object to be classified
Index	Integer	No	Index of the data object in the collection to be classified

Outcome outputs

Outcomes will be generated dynamically. The name of the outcome is configured for every classification. The output for such outcome is a data object of the classification category and type.

If the input data object is not of a category and type that the user is interested in, the node will return the outcome **Other** and now outputs will be passed out.

If the node classifies a data object within a collection, the triggered outcome will contain also output of type integer called **New Index** to be used for the test of the next object in the collection.

4.21 Delay

Model node type group	Control nodes
Palette	Control nodes

Description	This node component is intended to allow a branch of a process model to be delayed for a specified period of time. This node should only be used in workflow process models or other models that are run asynchronously: it is not allowed for any model called synchronously from an Operation to suspend. When a **Delay** node fires it will write a model resume record to the database providing the following information: • Model GUID • Node GUID • Resume Date/Time If a record exists for the Model and Node GUID combination, then the node will update the Resume date/time (in case the node is in a loop). Otherwise the node will insert a new record. The model resume table will be processed periodically (period determined by system configuration) by a service. The service will pick up all records that are on or past the resume date/time and resume the associated models. When the model is resumed, the model resume record will be deleted from the database. The amount of time for the delay can be set at run time. The input **Delay for** is an integer that can be set to the number of seconds the delay node is to remain suspended. If this input is set, it will override the configured delay.
Supersede	If superseded, a **Delay** node will cause an existing model resume record to be deleted based on the Model and Node GUIDs.
Label	NLS text for **Delay** concatenated with the period of the delay. The delay period will be shown as ‘to date/time’ or ‘for day(s) hh(s):mm(s)’ depending on the node configuration.
Verification Rules	None.
Validation Rules	Each node must have 1 arc entering and 1 arc leaving.
Node configuration	
To	This is a predetermined time to up to which the node will delay. This is likely to be an infrequently used option. When used, this is the date and time value that will be written to the model resume record. The node component will allow the time to be specified to the minute, although the accuracy of resumption will depend on the system-configured periodicity of the resume service. This value is ignored if a delay period is provided.
For	This is a period specified as a number of days, hours and minutes for which the node will delay from the moment that it is fired. At runtime the node component will this period to the exact date/time that the node fired in order to generate the resume date/time that will be written to the model resume record. Again, the accuracy of resumption will depend on the system-configured periodicity of the resume service.

Inputs

Name	**Type**	**Mandatory**	**Description**
Delay For	Integer	No	If this input is provided, the node will delay for this number of seconds (subject to the frequency of the resume service). This overrides the **To** and **For** values configured for the node.

Outcome outputs	None
None	

4.22 EDGE integration nodes

For technical information about the EDGE nodes, please refer to the *EDGE2020 implementation guide*.

4.23 End

Model node type	Control nodes

group	
Palette	Control nodes
Description	This node is used to end the execution path of a model. When this node is reached the model ends and any nodes in the model are deleted. For asynchronous nodes this may mean that a component will try to resume a model that no longer exists. In these cases the component will log an error. The node must be configured to pass an outcome value to the calling context. Associated with each outcome is a list of selected model context items. These items will be passed as outputs into the calling context. When the first end node is encountered in a model ALL other nodes within the model are superseded. As these nodes will be running on different paths (from the path that requested supersession) it is not possible to determine at which point these nodes will be superseded. Also note that only certain nodes support supersession. This behaviour applies to both top level models and to sub models whether executed synchronously or asynchronously
Label	Selected outcome name.
Verification Rules	An **End** node must have at least one Model Outcome associated with it.
Validation Rules	1 A Model must have at least one **End** node. 2 There should be at least one path from the **Start** node to each **End** node. 3 Each **End** node must have 1 arc entering and 0 arcs leaving. 4 Ensures that at run-time all model data output will be available at each **End** node.
Node configuration	
Outcome(s)	The outcome that should be passed to the calling context. Each outcome consists of a name and a system name. The name will vary with language. The system name will not. An End node may have more than one outcome.
Selected model context items	A list of model context items. The selected items will be passed to the calling context as outputs for the selected outcome.
Inputs	
None	
Outcome outputs	As configured and passed to caller
As configured	

4.24 Export document content

Model node type group	Document management
Palette	Document management nodes
Description	This node is used to make the Document Content, for example an image file, accessible to the Image Viewer. Normally a model using this node is called direct from the web page, and its configuration does not need to be modified by the modeller. This functionality is provided as a node for security reasons, to ensure that access to the Repository is made only through the CRM Server.
Label	Export document to file
Verification Rules	None
Validation Rules	The node must have one arc entering and two arcs leaving.
Node configuration	
None	

Inputs

Name	**Type**	**Mandatory**	**Description**
Document ID	String	Yes	The Portrait unique identifier for the document to be viewed.

Outcome outputs	**OK**

Name	Type	Mandatory	Description
Document ID	String		The Document ID is returned on an OK outcome.
Export location	String		The location the image file has been exported to.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are • Unknown error • An error code returned from the Repository

4.25 Filter by decision

Model node type group	Decisions
Palette	Decisions
Description	This node filters a collection by evaluating rules assigned to the specified decision type. Rules are assigned to decision types at run-time using administration tools (such as the Campaign Manager). Rule assignment is flexible and is made for one or more "decision items" (e.g. for an individual campaign prompt, for all prompts in a Campaign, for all prompts in campaigns of a particular category etc.). This node filters an input collection of "decision items" by firstly, determining which rules are applicable to each decision item through assignments to the specified decision type, and then by executing those rules. A rule failure (evaluates to false) causes all affected decision items to be omitted from the output collection, 'Decision item collection'. If 'Return unmatched data' is set to True, the failed items are inserted in the Diagnostics output, along with the rule that caused the item to fail. The values for rule variables are obtained from two types of data objects (as defined in the decision type in the configuration tree): 1, The "Data Source". This is a data object that is available to rules for all decision items. Rule variables referencing the data source will always use the same value irrespective of decision item. 2, The decision item itself. Rule variables referencing a decision item will always use a value taken from the decision item for which the rule is currently being evaluated.
Label	Filter by decision
Verification Rules	A decision type must be associated with the node. The decision type must exist in the configuration tree.
Validation Rules	The node must have one arc entering and two arcs leaving.
Node configuration	
Decision type	The type of decision determines which rules must be evaluated to perform the filtering. When a decision type has been selected, the decision type input will be removed from the node.

Inputs			
Name	**Type**	**Mandatory**	**Description**
Decision type	String	Yes	The type of decision being made. This input will be removed when a selection is made via the edit dialog.
Data Source	Data Object	Yes	The data source for rule variable values. Category and type as defined in the decision type.
Decision item collection	Data Object Collection	Yes	The collection of items to be filtered. Category and type as defined in the decision type.

Engagement action ID	String	No	Reserved for future use.
Return unmatched data	Boolean	No	Whether to return unmatched items along with the rule they failed against. Defaults to false.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Decision item collection	Data Object Collection	N/A	The filtered collection of items.
Diagnostics	Data Object Collection	N/A	The list of failed decision items and the rules they failed on. Only created if 'Return unmatched data' is set to True.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**

4.26 Format view of CBE

Model node type group	Portrait
Palette	Portrait nodes
Description	This node generates a data object collection of view attributes for the CBE data object provided. The CBE category and type is derived from the Category and type characteristics of the input data object. A data object of category **ViewAttributes** and type **ViewAttribute** is created for each attribute in the view summary definition. The value for each will be populated from the value on the CBE Instance data object. If the view name is invalid for the provided CBE instance object or any other error occurs the information is logged in event log of the CRM server and the FAIL outcome is triggered.
Label	Format View of CBE
Verification Rules	None
Validation Rules	The node must have one arc entering and two arcs leaving.

Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
CBE Instance	Data Object	Yes	Unclassified data object input providing data for the instance of the CBE to be displayed. Typically, the data object is sourced from a retrieve details DAT. It could also be sourced from an external source. Category and Type properties of the provided data object instance are vital for the node to resolve which attributes to look for.
View name	String	Yes	System name of the view. Typically the system name is retrieved from Retrieve CBE Views DAT.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
View attributes	Data Object Collection	N/A	Collection of data objects with category **ViewAttributes** and type **ViewAttribute**.
Outcome outputs	**Fail**		

Name	**Type**	**Mandatory**	**Description**

4.27 Display generated interaction

Model node type group	Portrait
Palette	Generated interaction nodes
Description	This node is used to insert a generated user interface point into a model. The user interface is generated by the Generated interaction component which will deliver an XML representation of relevant questions to the channel enabler which in turn will render these on the Generated interaction page. The Generated interaction definition will present all questions as potential inputs and outputs to the node. Outcomes are determined by the configuration of other Generated interaction elements. The mechanics of execution for a Generated interaction node are in other respects, similar to those of a custom interaction node described above.
Label	None.
Verification Rules	1 A Generated interaction Node must have a Domain associated with it. 2 The Domain must exist in the Configuration Tree.
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
Generated interaction	This is the Generated interaction domain that will be run. The selected domain will determine the nodes inputs, outcomes and outputs.
Inputs	
Determined by configuration	
Outcome outputs	Determined by configuration
Determined by configuration	

4.28 Get data object from collection

Model node type group	Portrait (**Collection iterator**)		
Palette	Data nodes		
Description	This node is used to help a model deconstruct a collection into single elements. These elements may be simple properties or data objects. Effectively this iterator outputs a single element identified by an index input parameter. Consequently the output type is the same as the type within the collection.		
Label	This should be taken from the node configuration. If no configuration is provided, this should default to **Get data from collection**.		
Verification Rules	None		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
Label	**Get data object from collection**(previously called **Collection iterator**)		
Inputs			
Name	Type	Mandatory	Description
Collection	Collection (of any type)	Yes	This is the collection that is to be processed. Given the nature of models, the node need not be used to iterate through a collection. It may simply be used to extract one element from it.

Index	Number	No (this is necessary to allow the technique of using 'Map to new' and not providing a value.)	This determines which row will be extracted. The index is a 1 based offset into the collection. The value must be >= 1. If not provided it will be assumed to be 1.
Direction	String	No	This determines in which direction the list should be processed. The values will be either **NEXT** or **PREVIOUS**.
Outcome outputs	**OK**		
Name	Type	Mandatory	Description
Collection Element	Same type as the collection elements	N/A	This is the element identified by the Index parameter. The content of the element will be determined by the type of collection.
New Index	Number	N/A	This is the index of the next row as determined by the direction parameter. If direction is NEXT then this value equals: Index + 1. Otherwise it is: Index – 1.
Outcome outputs	**Out of bounds**		
None	(This outcome is used when the Index input parameter is outside the bounds of the collection.)		

4.29 Get smart look-up value

Model node type group	General
Palette	Portrait nodes
Description	A smart look-up node is used to get the values associated for the given smart look-up(s) and criteria. Current node implementation supports multiple smart look-ups value selection. When a user selects smart look-up(s), NodeConfig adds all Criteria associated with the smart look-up as Input of the Node. To avoid conflict with same Criteria name(s) for more than one smart look-up, the input name of Node is created by combination of Criteria name and smart look-up name. Input Name : Criteria(Smart lookup). And adds Smart look-up name as Output of Node.
Label	NLS text for **Get smart look-up value**.
Verification Rules	There are no verification rules for this node.
Validation Rules	Each node must have 1 arc entering and a maximum of 2, minimum of 1 leaving.
Node configuration	
Available Parameters	This is a list box filled with the list of available Smart look-ups in the configuration tree. Users can move available smart look-ups to selected smart look-up(s) and vice-versa using the button provided and the direction of move.
Selected Parameters	This list box will display all selected smart look-ups. On selection of a new smart look-up all criteria associated with the smart look-up will be added as Node inputs, and the smart look-up name as node output. On deselect all inputs and output for that smart look-up will be removed from the Node input/output list.

Inputs			
Name	**Type**	**Mandatory**	**Description**
Inputs to node(s) are criteria associated with smart look-ups	Criteria Type	No	
Outcome outputs	**Match**		
Name	**Type**	**Mandatory**	**Description**
Name of Node Outputs are same as name of associated smart look-ups	Streamable	Yes	Output of Node is always IamcResultSet object (Streamable), which contains the list of cell value name (may be NULL) and value.
Outcome outputs	**NotMatch**		
None			
No output will be generated for this outcome			

4.30 Get smart lookup value 2

Model node type group	General
Palette	Portrait nodes
Description	A smart lookup 2 node is used to get the values associated with given smart look up(s) for the supplied criteria. The node allows multiple lookups on the same or different smart lookups. When the user adds a smart lookup to the node's configuration an input is added to the node that is the criteria data object for that smart lookup. Similarly an output is added to the node on the *Match* outcome for that smart lookup. The user has the option of consolidating into one collection the output of the smart lookups configured in the node; in this case no additional output is generated. To avoid name conflicts the user can enter an alias for each smart lookup added to the node. This alias is used as the prefix of the input and output of the smart lookup. The suffix "_Criteria" is added to the alias for the input and the suffix "_Results" is added to the alias for the output.
Label	NLS text for Get smart lookup value 2.
Verification Rules	There are no verification rules for this node.
Validation Rules	Each node must have 1 arc entering and a maximum of 2, minimum of 1 leaving.

Node configuration			
Edit smart lookup 2 dialogs	The initial dialog of the node shows the list of smart lookups currently configured. The user can add, edit and remove smart lookups in the list. A second dialog is used for adding and editing the smart lookups. In this dialog the user can select from a list of all available smart lookups, define an alias name and select to consolidate the results.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
Inputs to the node are the criteria data objects for the smart lookups.	Data object	No	The data object will be the following type. Category = "SmartLookupCriteria" Type = "Smart lookup system name".
Outcome outputs	**Match**		

Name	Type	Mandatory	Description
The outputs will be the results of the smart lookups configured in the node.	Collection if the smart lookup is multi valued. Data object if it is single valued.	N/A.	The data object will be the following type. Category = "SmartLookupValue" Type = "Smart lookup value type".
Outcome outputs	**Not match**		
None			

4.31 Get task

Model node type group	Workflow nodes
Palette	Workflow nodes
Description	This node component is used to retrieve a task from the Task Engine alongwith TaskHistory data (optional) based on the inputs provided. The node component allows explicit retrieval of a task using a specified Task ID or the task engine can choose the task. The task chosen to retrieve is the first task from the set of tasks fulfilling the search conditions, which is ordered by task priority, related party relationship level and task creation date. In other words, the task with highest priority is retrieved. If there are multiple such tasks, the one with the highest related party relationship level is retrieved. If there are still multiple tasks, the oldest task from them is retrieved. The Task Engine does not provide any task permission handling. Its primary function is to retrieve the most appropriate task in the queue. Permissions are handled through the parameterisation component. In order to retrieve only those tasks that the user is permitted to perform, the calling model must retrieve all the permissible task types prior to invoking this node component. The permissible task types are then passed in as a collection. The collection is ignored if an explicit identifier is provided. The only status that a task can have that makes it ineligible for selection is **Checked Out**. By default, if the node finds a task successfully, it will check out the task to the current user. But it is possible to change this behaviour by using the **Checkout task** input.
Label	Locale specific version of **Get task**.
Verification Rules	There are no verification rules for this node. The Task ID and Party Type are supplied by the Input Model at runtime.
Validation Rules	Each code node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
None	
Inputs	

Name	Type	Mandatory	Description
Party ID	String	Yes	Identifier of the user performing the get. This identifier is used to identify the owner of the checked-out task.
Party Type	String	Yes	This is the system name of the party that Party ID is an instance of.
Task ID **(Instance ID)**	String	No	This is the identifier of a specific task. Typically, this task will have been chosen by the **Pick Work** dialog. The **Pick Work** dialog must ensure that the user has permission to get the task. If this parameter is not supplied, the node component will invoke the task engine's selection algorithm to retrieve an active task from the queue.

Checkout task	Boolean	No	Determines if the task returned by the node is left in a checked out state. The task will not be checked out if this value is set to False. By default this value is set to True and the task is checked out to the current user.
Permissible task **(Task Permissions)**	Data Object Collection	No	This is a collection of task type system names. The collection of names will typically be retrieved using a **Get Smart look-up value** node. The data object will contain just one property: Task type system name. If omitted and the Task ID has also been omitted then the node component will return tasks of any type.
IncludeTaskHistory	Boolean	No	If true then the results will include task history collection. Use false for better performance. If no value is provided, it defaults to 'True'
TaskHistoryMaxRowCount	Integer	No	The maximum number of task history records to retrieve for each task instance. This input will only be used when 'Include task history' is set to true. The default value is 1000.

Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Task instance	Data object	Yes	This is the data object that contains properties for the chosen task instance. The data object is described in more detail at the start of this section.

Outcome outputs	**Fail**
None	

Outcome outputs	**No Work**
None	

4.32 Retrieve GI answers

Model node type group	Portrait
Palette	Generated interaction nodes
Description	This node component is used to retrieve the answers to questions in the identified generated interaction session. Typically, the answers will be used to populate a combo box in order to answer another question. The node component will support retrieval of answers based on the Answer ID of an answer, the qualifying path of an answer or the answer(s) to any question(s) that match the keywords supplied.
Label	Locale sensitive version of **GI Retrieve answers**.
Verification Rules	None.
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.

Node configuration
None

Inputs			
Name	**Type**	**Mandatory**	**Description**
DQE Session ID	String	Yes	This is the identifier of a currently active session. The node will not retrieve answers for a session that has completed. If no other parameters are provided, the node will return all the answers in the session.

Answer ID	String	No	This is the system name of the answer as configured in the question definition. Since the same Answer ID may be used in a number of answers with different qualifying paths, use of this parameter may produce a number of answer values. This parameter is ignored if the Qualifying path is provided. The values that satisfy this criterion may be further filtered by the Keywords parameter.
Qualifying path	String	No	This is the full qualifying path of an answer. Where appropriate, the Qualifying path may include the index of an answer to a multipliable question. If the index value supplied is an asterisk (*), then all the answers to the multipliable question will be returned. If this parameter is provided, both the Answer ID and Keywords parameters are ignored.
Keywords	String	No	This is a comma-separated list of keywords. The keywords may include the reserved keywords: Relevant Irrelevant Answered Unanswered Mandatory Optional The answers that are found are answers to questions that have been configured with all of the keywords provided in this parameter irrespective of order. This parameter may be used in conjunction with the Answer ID parameter. Keywords specified for a Compound question are applied to all of the question's compound-able children in addition to any keywords they have defined. The same applies to sub-domains.
Outcome outputs	**OK**		
Name	Type	Mandatory	Description
Answer list	Data object collection	Yes	This is the collection of matching answer values. Each data object contains the following properties: Answer value (display) Answer value (actual) Qualifying path
Outcome outputs	**Fail**		
None			

4.33 Retrieve GI view details

Model node type group	Portrait
Palette	Generated interaction nodes

Description	This node component is used to retrieve all the elements in a specified view. The node component will typically be used to support a data access model (DAM) as part of a custom control in order to render the contents of the view in a generated interaction session. A view is a subset of the configured elements of the generated interaction domain and can bring together elements that are otherwise configured on different pages. The node component reads the keywords configured in the selected view and proceeds to search through all the elements in the domain that have also got all of the views keywords configured. Certain keywords are reserved to match on the status of an element. These are: • Relevant (relevance status) • Irrelevant (relevance status) • Answered (answered status) • Unanswered (answered status) • Mandatory (mandatory status) • Optional (mandatory status) The node component will include text and variable elements in its search, but: • For text elements it will ignore the answered and mandatory status keywords. • For variable elements it will ignore the relevance and mandatory keywords.		
Label	Locale sensitive version of **GI retrieve view details**.		
Verification Rules	1 Each node must have a Domain Name set. 2 The Domain name must exist in the Configuration Tree. 3 Each Node must have a Domain View set. 4 The Domain View must exist in the Configuration Tree.		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
DQE Domain	This is a combo box of generated interaction domains. The configurer will be required to select from a list of domains configured into the <project>.amc file. The list will not include compound domains.		
View name	This is a combo box of views configured for the selected domain. No selection will be possible until a domain has been chosen. If no views exist, it will not be possible to configure the node successfully.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
DQE Session ID	String	Yes	This is the identifier of a currently active session. The node will not retrieve view details for a session that has completed.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
View elements	Data Object Collection	N/A	This is the dynamic, heterogeneous collection of elements that have been retrieved for the view. At design time it will not be possible to know the contents of the collection, consequently it will be necessary to map the entire collection as an output rather than trying to map out individual elements. Each data object will contain the properties relevant to the element type.
Outcome outputs	**Fail**		
None			

4.34 Access globals

Model node type group	Portrait
Palette	Portrait nodes

Description	The purpose of this node is to allow a modeller to map data to and from the Global Data Context. Normally, the Global Data Context is read from and written to by Operations. However, in certain circumstances (for example custom controls) it may be necessary for non-operation models to access global data. This node component will list items from the Global Data Context as input and output parameters and thus allow a modeller to map these to items in the model context.
Label	NLS text for **Globals**.
Node configuration	
None	
Inputs	
Items in the Global Data Context. These are read from the **Globals** section of the project's workspace file.	
Outcome outputs	**OK**
Items in the Global Data Context. These are read from the **Globals** section of the project's workspace file.	

4.35 Loop end

Model node type group	Control nodes
Palette	Control nodes
Description	This node is used to specify the point at which a model can return to re-run a previous segment of the model in a loop. Ultimately, the **Loop end** will connect to a corresponding **Loop start**. Together, the two nodes define the loop brackets. The modeller can add nodes between the **Loop end** and **Loop start** nodes. Using configuration, the modeller can specify a maximum loop count.
Label	If a maximum lop count is configured, this should be the NLS text for **Limit** = + configured limit. Otherwise this should be the NLS text for **Unlimited**.
Verification Rules	None.
Validation Rules	1 **Loop start** nodes and **Loop end** nodes counts should be equal. 2 Each **Loop end** node must lie on a closed path. 3 Each **Loop end** node must have 1 arc entering and 1 or 2 arcs leaving.
Node configuration	
Loop limit	Specifies the maximum number of times through the loop. If the loop limit is exceeded (that is the node has fired as often as this configured value) then the node will produce a **Don't loop** outcome. Otherwise the node will produce a **Loop** outcome. If the value is not configured the node only provides the Loop outcome.
Inputs	
None	
Outcome outputs	**Loop**
None	
Outcome outputs	**Don't Loop**
None	

4.36 Loop start

Model node type group	Control nodes
Palette	Control nodes

Description	This node is used to specify the start of a model segment that can be executed in a loop. The node is used in conjunction with the **End** Loop node to define the loop segment. Optionally, the modeller can map the loop counter value into the model context for use by other nodes. The **Loop** nodes do not use the context value and consequently, no node can influence the loop count as far as the loop nodes are concerned. A **Loop start** will automatically find its associated end loop by searching the model using the following rules: 1 It will navigate along its input paths, testing each one in turn until it finds the one that leads to start node. 2 Ignoring the path found in 1) it will navigate its input paths until it finds an **End loop** node that has the **Loop** outcome on the current path. Any other **End loop** nodes will be ignored. The model will only be valid if each **Loop start** corresponds to one and only one **Loop end** node.
Label	None.
Verification Rules	If a Loop start node outputs its counter to the Data Context, it must do so using a Data Context Entry of type Long.
Validation Rules	1 Loop start nodes must be equal in count to **Loop end** nodes. 2 Each **Loop start** node must lie on a closed path. 3 Each **Loop start** node must have 2 arcs entering and 1 arc leaving. 4 Loop start node must be reachable from the Model's **Start** node. 5 A loop can only be entered by going through the **Loop start** node. 6 A **Loop start** node must be associated unambiguously with a single **Loop end** Node.

Node configuration

None

Inputs

None

Outcome outputs

Name	Type	Mandatory	Description
Counter	Integer	No	This is the current value of the loop counter. The first time that the loop start node fires, the loop counter is set to 1. Thereafter it is incremented by 1.

4.37 Run model via class

Model node type group	Portrait
Palette	Portrait nodes
Description	This node is used to execute a submodel selected in runtime that supports a specific interface defined via configuration. The model class is configured with model class inputs, outcomes, outputs and class members. Class members are configured as a list of candidate sub models, from which one will be selected to execute at runtime. Every class member has configured input, outcome and output mappings between the submodel interface and the class interface. Class model availability criteria determine, which class member can be selected for execution at runtime. The input '**Class Member**', system name, is then used to determine, which submodel to run. If this system name does not match any member then the '**Not A member**' outcome is triggered. Once a submodel to run is selected to run, the inputs to the model are created by processing the mappings created at configuration time for the selected member. The submodel is than run to its completion. As well as a submodel node, the model class node is configured to run in one of two modes - synchronously (wait for the submodel to complete) or asynchronously (do not wait for its completion).

	In asynchronous mode, the outcome **‘Model Started’** (or **‘Not a Member’** outcome) is triggered immediately. Whereas in synchronous mode, once the submodel has completed, the mappings created at configuration time are used to produce the outcomes/outputs for the class node from the outcome/outputs of the submodel. If a submodel outcome cannot be resolved back to the model class outcome or if any other error in mapping occurs, the node resumes the model with an outcome of **‘Not a Member’** and its associated outputs (**‘Display name’**, **‘System name’** and ‘**Version number**’).
Supersede	When superseded, the submodel will unconditionally superseded all nodes within the submodel. If the submodel itself contains sub models, these too will be superseded. As these nodes will be running on different paths (from the path that requested supersession) it is not possible to determine at what point these nodes will be superseded. Also, note that only certain nodes support supersession. For example, sub models that are run asynchronously do not support supersession.
Label	The name and version of the model class node.
Verification Rules	1 A model class must be associated with the node. 2 The model class must exist in the configuration tree.
Validation Rules	Same as for the submodel node.
Node configuration	
Model class	This is the selected model class. This selection will contribute to the determination of the inputs, outcomes and outputs of the submodel and the model class node.
Description	Once the model class has been selected, its description is displayed in read-only form so that the configurer is informed of the function of the selected model class.
Wait for model to complete	This is the selected mode of model class node execution. Select **True** so that at runtime, the parent model will wait until the submodel (associated with the selected model class) has completed before continuing. That is, the submodel will execute synchronously. In this mode, the sub model’s outcomes and outputs (after execution) will be mapped to the outcomes and outputs of the model class node. Select **False** so that at runtime, the parent model (and therefore the model class node) will continue without waiting for the submodel to complete. This is an asynchronous mode. Note that after the submodel is executed asynchronously the outcome of the model class node will always be **ModelStarted**. There will not be any mappings of the outcomes or outputs.

Inputs			
Name	Type	Mandatory	Description
Class member	String	Yes	The system name that has been defined for the model class. In other words, it is the system name of the submodel that will be selected to run at runtime.
Other inputs			Determined by the configuration of the model class, i.e. its explicit interface. All of these inputs will be mandatory, if via model usage mapping, they are mapped onto mandatory inputs of the submodel. Otherwise, these model class node inputs will be optional.
Outcome outputs	Model started		

Outcome outputs	**Not a member**		
Name	**Type**	**Mandatory**	**Description**
Display name	**String**		**Display name**
System name	**String**		**System name**
Version number	**String**		**Version number**
Outcome outputs	Other		

Name	Type	Mandatory	Description
			If the 'Wait for model to complete' input is set to True, then the model class node will wait until the submodel is completed and map the outcomes/outputs (in accordance with its configuration).

4.38 Or

Model node type group	Control nodes
Palette	Control nodes
Description	This node is used to manage the combination of dependency arcs. The node waits until the first of the connected nodes completes with the specified outcomes.
Label	NLS text for **Or**.
Validation Rules	Each Or node must have >=1 Arc entering and 1 arc leaving.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A
None	

4.39 Publish change notifications

Model node type group	General
Palette	Portrait
Description	The node creates a set of change records in the session context. The set is chosen by the node editor from those available in **Supporting definitions\Change notification catagories**. Selected change categories that require parameters to fully specify them cause inputs to be exposed on the node, all of which are mandatory. There are no outcomes from the node.
Label	**Publish change notifications**
Verification Rules	Any inputs defined by change parameters must be mapped.
Validation Rules	The node must have one arc entering and one arc leaving.
Node configuration	
Selected change categories.	
Inputs	
Variable, depending on the selected change categories and the parameters declared thereon.	
Outcome outputs	
None	

4.40 Put back task

Model node type group	Workflow nodes
Palette	Workflow nodes

Description	This node component is used to return the current task to the active pool task in its original state. The only change to the task is that the task history is updated to reflect the put back action. Only the owner (the user that has the task checked out) can put the task back. The task can only be put back if it is checked out!		
Label	Local sensitive version of **Put back task**.		
Verification Rules	There are no verification rules for this node. The Task ID is supplied by the Input Model at runtime.		
Validation Rules	Each code node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Task ID (Instance ID)	Yes	Yes	This is the identifier of the task to put back.
Party ID	String	Yes	This is the identifier of the user putting the task back. This must be the same as the identifier of the user that has the task checked out. In cases where tasks remain checked out and blocked, they may be put back by a system administrator using an administrator's function.
Party Type	String	Yes	This is the system name of the party that Party ID is an instance of.
Comments	String	No	Notes that should be added to the task history. The task history should also be updated to show the date and time that the task was put back as well as the Party ID.
Outcome outputs	**OK**		
Name	Type	Mandatory	Description
Outcome outputs	**Fail**		
None			
Outcome outputs	**NoAccess**		
None			

4.41 Reach milestone

This node is used to update a milestone in the context of a Business Process.

Program Identifier	**AIT.AMC.Node.ReachMilestoneConfig.1**
Model node type group	General
Palette	Case
Description	This node component is responsible for updating a milestone instance in the database to indicate that the milestone has been reached. Two mandatory values must be provided at update time: the business process type and business process id. The node makes the following checks: • Whether the supplied business process type is a member of the **BusinessProcessType** reference data group. The supplied business process id is then checked against the appropriate primary key to test its validity e.g. if the type is **Case** then the **amc_pce_case.case_id** column is checked. • Whether a milestone instance exists of the configured milestone type within the supplied Business Process. If one does not exist, the **Milestone not found** outcome is fired. • Milestone type is also a mandatory input. However, this can be selected either at configuration time; by selecting the milestone type via the Edit dialog, or mapped as a node input.

Label	NLS text for **Reach milestone**.		
Verification Rules	None.		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
Milestone type	The type of milestone to be updated checked against the value in the Milestone input and the **MilestoneType** reference data group. When a milestone type has been selected, the Milestone type input will be removed from the node.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
Milestone type	String	Yes	The type of milestone being reached. This input will be removed when a selection is made via the Edit dialog.
Milestone	DataObject	No	Details of the milestone that has been reached. Category: Milestone Type: Milestone If the milestone that has been reached has status 'Not reached', then the following milestone attribute updates are made: • **MilestoneStatus** is set to Reached. • **ReachedBy** is set to the corresponding property value supplied in the **Milestone** data object. • **ReachedWhen** is set to the current date and time. • Configured milestone attributes values are set to those supplied in the **Milestone** data object. If the milestone that has been reached does not have a status of 'Not reached', then a new milestone is created using the values supplied in the Milestone data object. In the new milestone: • MilestoneStatus is set to Duplicate • **ParentMilestoneId** is set to the id of the milestone that has been reached • **InstanceId** is incremented to ensure uniqueness amongst duplicate milestones. E.g. when a milestone has been reached for the third time, then the **InstanceId** in the duplicate milestone will be set to 2.
EngagementAction Id	String	No	• The unique identifier of the engagement action that has been created when this milestone was reached.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Milestone Id	String	No	The Id of the Milestone updated (or the new milestone in the case where a Milestone has been reached more than once)
Outcome outputs	**Fail**		
None			
Outcome outputs	**Milestone not found**		
None			

4.42 Remove document associations

Model node type group	Document management
Palette	Document management nodes

Description	This node is used to correct errors in association, by removing incorrect associations to Parties and Contracts. The inputs can either pass in a collection of incorrect associations to be removed, or the inputs can request All Parties or All Contracts to be removed. If the removal of any association fails, the whole transaction fails and no associations are removed.		
Label	Remove document associations		
Verification Rules	There must be at least one entity association to be removed, so at least one of the following inputs must be mapped. • All Parties • All Contracts • Parties • Contracts		
Validation Rules	The node must have one arc entering and two arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Document ID	String	Yes	Unique identifier of the document from which associations are being removed.
All parties	Bool	No	If True then all party associations are to be removed.
All contracts	Bool	No	If True then all contract associations are to be removed.
Parties	Collection	No	A collection of Party Reference data objects for the associations which are to be removed. The Party Reference data object properties are as below. • Party ID String. This is mandatory.
Contracts	Collection	No	A collection of Contract Reference data objects for the associations which are to be removed. The Contract Reference data object properties are as below. • Contract ID String. This is mandatory.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Document ID	String		The node returns the Document ID if the requested references have been removed successfully.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible Fail codes are as follows. • Invalid references. One or more of the references passed in is invalid. The Invalid References are returned in the Invalid References collection. • References collections • Empty inputs. All the collections passed in were empty, and All Parties and All Contracts were both False. No associations were removed. No Invalid References collection is returned. • Unknown error. The transaction has failed for an unknown reason. No associations have been removed. No Invalid References collection is returned.

Invalid references	Data object collection		If any of the references are invalid, the node will return an Invalid References collection. This contains only the invalid references. The Invalid Reference data object has the following structure. • Object type (one of Party or Contract) • Object ID

4.43 Resume task

Model node type group	Workflow
Palette	Workflow nodes
Description	This node component is used to resume a suspended task and returns the suspended task to an active task status.
Label	Configured document name.
Verification Rules	There are no verification rules for this node. The Task ID, Party ID and Party Type are supplied by the Input Model at runtime.
Validation Rules	Each code node must have 1 arc entering and >= 1 arcs leaving.

Node configuration

None

Inputs

Name	Type	Mandatory	Description
Task ID	String	Yes	This is the identifier of the task, which needs to be resumed.
Party ID	String	Yes	The identifier of the user resuming the task. This must be the same as the identifier of the user that has the task checked out.
Party Type	String	Yes	This is the system name of the party that ‘Party ID’ is an instance of.
Comments	String	No	Notes that should be added to the task history. The task history should also be updated to show the date and time that the task was put back as well as the Party ID.

Outcome outputs **OK**

Name	Type	Mandatory	Description
Task instance	Data object	Yes	The data object containing details of the newly resumed task.

Outcome outputs **Fail**

None

Outcome outputs **NoAccess**

None

4.44 Retrieve composite data object

Program Identifier	**AIT.AMC.ProcessEngine.RetrieveCDONode.1**
Model node type group	Portrait
Palette	Portrait
Description	This node is used to retrieve a composite data object.
Label	Retrieve composite data object
Validation Rules	Each node must have 1 arc entering and 2 arcs leaving.

Node configuration

Composite data object	The type of composite data object to be retrieved. If no type is selected the node can be used to retrieve any type of composite data objects.		
Enable data access classing	If selected, 2 additional inputs are added to the node to allow the use of data access classes. An additional outcome “Not a member” is also added.		
Retrieve only element IDs	Check this box to retrieve only the ID property of each element within the composite data object. This simply returns the structure of the composite data object and populates all the elements with their unique identifier.		
Return newly defined objects as empty	By default all objects that are found in the CDO definition, but not retrieved from the instance data in the database are returned as Null objects with no properties. Check this box if you wish these objects to be returned as populated object with empty properties.		
Return null saved objects as empty	By default all CDO objects that are Null when they are saved, get returned as Null objects with no properties. Check this box if you wish these objects to be returned as populated object with empty properties.		
Inputs			
Composite data object ID	String	Yes	The unique identifier of the composite data object to retrieve.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Composite data object	Data object	N/A	The retrieved composite data object.
Outcome outputs	**Fail**		
None			

4.45 Retrieve document

Model node type group	Document management		
Palette	Document management nodes		
Description	The node returns a document data object for the requested ID.		
Label	Retrieve document		
Verification Rules			
Validation Rules	The node must have one arc entering and two arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Document ID	String	Yes	Unique identifier of the document being retrieved.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Document	Data object		The document requested.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are: • Unknown error

4.46 Retrieve envelope

Model node type	Document management

group			
Palette	Document management nodes		
Description	The node returns an envelope data object for the ID given. Note than an Envelope data object always contains at least one Document.		
Label	**Retrieve envelope**		
Verification Rules			
Validation Rules	The node must have one arc entering and two arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Envelope ID	String	Yes	Unique identifier of the envelope being retrieved.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Envelope	Data object		The envelope requested.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are: • Unknown error

4.47 Generate menu

Model node type group	Portrait		
Palette	Portrait nodes		
Description	The retrieve menu definition node is used to retrieve configured menu definitions, and optionally filter menu items when smart lookup results are supplied. The user can choose the fix the menu definition retrieved, or specify the menu system name at runtime. Any number of input collections of smart lookup results can be specified to allow the node to filter the menu items based on the results of a smart lookup. If the user specifies that no input collections are to be used, then no filtering of the menu items is performed, and the complete menu definition is retrieved.		
Label	NLS text for Retrieve menu definition.		
Verification Rules	There are no verification rules for this node.		
Validation Rules	Each node must have 1 arc entering and a maximum of 2, minimum of 1 leaving.		
Node configuration			
Edit smart lookup 2 dialogs	The dialog allows the user to choose the menu definition to retrieve from a combo box or specify that the value should be supplied at runtime. The number of input collections can be specified to allow menu item filtering.		
Inputs			
Name	**Type**	**Mandatory**	**Description**
Menu system name	String	Yes (if required)	This is the system name of the menu item to retrieve. The input is only added if the user specifies to choose the menu definition at runtime.

Input collection *X*	Data object collection	No	The name of the input will be suffixed by the number of the collection. The data objects in the collection must be the results of a smart lookup. Non smart lookup results and results that do not correspond to any menu item are ignored. It is valid for the collections to contain no objects.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Menu definition XML	String	N/A.	The XML representation of the menu definition. A scheme definition of the XML is available in the Portrait SDK.
Outcome outputs	**Invalid menu name**		

This outcome is only available if the menu definition is specified at runtime.

None

4.48 Save composite data object

Program Identifier	**AIT.AMC.ProcessEngine.SaveCDONode.1**
Model node type group	Portrait
Palette	Portrait
Description	This node is used to save a composite data object.
Label	Save composite data object
Verification Rules	None.
Validation Rules	Each node must have 1 arc entering and 2 arcs leaving.
Node configuration	
Composite data object	The type of composite data object to be saved. If no type is selected the node can be used to save any type of composite data object.
Enable data access classing	If selected, 2 additional inputs are added to the node to allow the use of data access classes. An additional outcome “Not a member” is also added.
Check if modified	Used for locking. An additional outcome “Object has been modified” is also added.

Inputs			
Name	**Type**	**Mandatory**	**Description**
Composite data object	Data object	Yes	The composite data object to be saved.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Composite data object ID	String	N/A	The unique identifier of the saved composite data object.
Outcome outputs	**Fail**		

None

4.49 Save document

Model node type group	Document management
Palette	Document management nodes
Description	The node updates an existing Document data object. This node is used for correction of errors, for example, to update Status in Send for Re-Scan.

Label	**Update document**		
Verification Rules	None		
Validation Rules	The node must have one arc entering and two arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Document	Data object	Yes	A Document data object.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Document ID	String		The ID of the document being updated.
Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are: • Unknown error • Invalid Envelope ID – that is, the Envelope ID does not exist. • Invalid Document – that is, either there is no Document ID, or the ID does not exist.

4.50 Save envelope

Model node type group	Document management		
Palette	Document management nodes		
Description	The node either saves a new envelope data object, or updates an existing one. The input to the node is an Envelope data object. If no Envelope ID is provided the node creates a new Envelope using the data in the Envelope data object. If an Envelope ID is provided the node updates the Envelope for the given ID. Any Documents contained in the Envelope are added to the Envelope. If the Document has no ID then a new Document record is created and it is added to the Envelope. If the Document has an ID then it is removed from the Envelope to which it is currently attached and added to this Envelope. If the Document already belongs to this Envelope then it is ignored (even if it contains different data). Documents can only be added to or removed from an Envelope, using Save Envelope. They must be updated using Update Document. Note that Document records can only be created using Save Envelope. A valid Document record must always belong to an Envelope.		
Label	**Save envelope**		
Verification Rules	None		
Validation Rules	The node must have one arc entering and two arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Envelope	Data object	Yes	A valid Envelope data object.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Envelope ID	String		The ID of the envelope being created or updated.

Outcome outputs	**Fail**		
Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are: • Unknown error • Invalid Envelope (see Envelope data object definition)

4.51 Save task

Model node type group	Workflow nodes
Palette	Workflow nodes
Description	The **Save task** node may be used from within an operation model in order to save the attributes of task. This will only be necessary if the value of an attribute has changed and the change is needed at a later stage in the workflow process model (WPM).
Label	**Save task**
Validation	The **Save task** node will take a Task Type Data Object as its input. The Data Object will contain the Task ID fixed property, which must be set for the save to take place. The save will only succeed on a task that is checked out to the user performing the task. If status of the task was not **active**, OR **passive** OR **suspended**, the node will not complete the task and will reach **NoAccess** outcome instead.

Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
TaskInstance	DataObject	Yes	Contains TaskAttributes being saved.
PartyID	String	Yes	Party performing SaveTask operation.
PartyType	String	Yes	Party performing SaveTask operation.
Description	String	No	Notes that should be added to the task history.
Outcome outputs	**OK**		
None			
Outcome outputs	**NoAccess**		
None			
Outcome outputs	**Fail**		
None			

4.52 Search for documents

Model node type group	Document management
Palette	Document management nodes

Description	The node returns a collection of document data object for the search criteria given. The search criteria supported are as follows. • Party collection. All documents which are associated to a member of a collection of parties. • Contract collection. All documents which are associated to a member of a collection of contracts. If no documents are found the node returns an OK outcome with an empty collection. If both Parties and Contracts Search Criteria are provided, the search is made using both criteria, that is, the search returns all documents that have an association to one of the Parties AND to one of the Contracts. If an empty collection is provided as an input, the node returns an OK outcome with an empty collection. If no input is provided the node returns a Fail outcome. If any of the references provided are invalid, that is, the Party ID or Contract ID does not exist, the node returns an OK outcome with an empty collection.
Label	**Search for documents**
Verification Rules	There must be at least one search criterion mapped to the inputs.
Validation Rules	The node must have one arc entering and two arcs leaving.

Node configuration

None

Inputs

Name	**Type**	**Mandatory**	**Description**
Parties	Collection	No	A collection of Party Reference data objects.
Contracts	Collection	No	A collection of Contract Reference data objects.

Outcome outputs **OK**

Name	**Type**	**Mandatory**	**Description**
Documents	Collection		A collection of document data objects.

Outcome outputs **Fail**

Name	**Type**	**Mandatory**	**Description**
Fail code	String		The possible values are Unknown error No search criteria

4.53 Send all from outbox

Model node type group	Simplex channel
Palette	Simplex
Description	This node component is used to send all envelopes with attached documents, if supplied, that reside in the simplex channel outbox for a particular session identifier. When the **Send Immediate** input to the Create outbound document node is set to FALSE, this Send all from outbox node will generate all documents for the supplied session as a batch with a single cover letter.
Label	Send all from outbox.
Verification Rules	1 An **Engagement Session ID must be specified**.
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.

Node configuration

None

Inputs

Name	Type	Mandatory	Description
Engagement Session ID	String	Yes	This is the identifier of the current engagement. The **Start Engagement** DAT will return this value.

Outcome outputs	**OK**
None	

Outcome outputs	**Failed**
None	

4.54 Send client event

Model node type group	General nodes
Palette	Portrait nodes
Description	This node is used to trigger a client event. Current Event Subscriptions are searched and every client that has subscribed to the particular Event Type and Event ID will be sent (asynchronously) the event. The node also caters for a payload to be sent, which is a string that can contain any data format that will be understood by the browser application.
Label	Send Client Event.
Verification Rules	None.
Validation Rules	1 Each **Send Client Event** node must have 1 arcs entering and 1 or 2 arcs leaving.

Node configuration

None

Inputs

Name	Type	Mandatory	Description
EventType	String	Yes	The event Type to send. This is the SystemName of the Reference Data item in the ClientEvents Reference Data Group. If the item is not defined, it is processed using the default Subscriber and Resolver. It is good practice to make sure all event types are defined in reference data, as this will prevent unnecessary errors being reported when names are not found in reference data
EventID	String	No	This is the qualifying data for the event Type. Typically, this would refine the Event Resolution process. For example, if the Event Type was “Consumer”, the event ID may well be the Consumer ID. **NB:** EventID should normally be provided, even if an empty string, as JavaScript limitations in the browser can prevent NULL ID’s being used.
EventPayload	String	No	Any free format data pertinent to the event
Synch	Boolean	No	Determines whether the Client Event manager will process the Send Event Resolution process synchronously or not. If yes, then the result of the node reflects the success of this process or not. The actual sending of the vent is ALWAYS processed asynchronously.

Outcome outputs	OK
None	

Outcome outputs	Fail
None	

4.55 Start

Model node type group	Control nodes
Palette	Control nodes
Description	This node is used to define the start of a model.
Label	None.
Verification Rules	None.
Validation Rules	1 A Model must have exactly one **Start** node. 2 Each **Start** node must have 0 arcs entering and 1 arc leaving.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A
None	

4.56 Run model

Model node type group	Portrait
Palette	Portrait nodes
Description	This node is used to run a model within a model. A submodel can be run synchronously (that is the parent model will wait until the submodel has completed before continuing execution) or asynchronously. The inputs, outcomes and outputs of this node are determined entirely by those of the submodel. The inputs of any model are deduced from items in the model's context that are mapped as inputs to a node but have not been mapped as outputs from a previously executing node. When superseded, a submodel node will unconditionally superseded all nodes within the submodel. If the submodel contains sub models, these too will be superseded. As these nodes will be running on different paths (from the path that requested supersession) it is not possible to determine at which point these nodes will be superseded. Also note that only certain nodes support supersession. Sub models that are run asynchronously do not support supersession.
Label	Name and version number of the referenced submodel.
Node configuration	
Sub model	This is the identifier (GUID) of the latest version of the submodel. At least it should be. In fact this is the GUID of the model that exists in the current configuration tree. The submodel is selected from a list which contains the model's name. Once the Model is specified, the node reads its input, outcome and output specification.
Wait for sub model to complete	Select **Yes** and the parent model will wait until the submodel has completed before continuing (that is the submodel is executed synchronously). In this situation the sub model's outcomes and outputs are the outcomes and outputs of the node. Select **No** and the model will continue without waiting for the submodel to complete (asynchronously). Also, when the submodel is executed asynchronously the outcome of the submodel node will always be **Model Started**.
Inputs	
Determined by configuration	
Outcome outputs	Determined by configuration if submodel executed synchronously (Wait for submodel to complete = YES), otherwise the only outcome is fixed as 'Model Started'.

Determined by configuration

4.57 Supersede

Model node type group	Portrait
Palette	Control nodes
Description	When fired, this node causes all the configured nodes to be superseded. In fact this node calls the Supersede interface on each of the configured nodes.
Label	NLS text for **Supersede**.
Verification Rules	None
Validation Rules	1 Each **Supersede** node must have 1 arc entering and 1 arcs leaving. 2 A **Supersede** node cannot supersede a node that lies on a path between itself and an **End** node. 3 A **Supersede** node cannot supersede a node that lies in a loop that is on a path between itself and an **End** node. 4 A **Supersede** node cannot supersede a node that lies in the same And/Branch network or which would stop an **And** node that lies between itself and an **End** node from completing.
Node configuration	
Nodes to supersede	The configuration is the list of nodes that will be superseded and this is selected from the nodes in the current model.
Inputs	
None	
Outcome outputs	N/A
None	

4.58 Suspend task

Model node type group	Workflow nodes		
Palette	Workflow nodes		
Description	This node component is used to suspend the specified task. Active tasks that can be suspended by agents with the configured availability criterion levels will have a **Suspend task** item in their **Task Summary**'s action menu. Tasks are required to be checked out prior to this operation and will be put back as a consequence.		
Label	Locale sensitive version of **Suspend task**.		
Verification Rules	There are no verification rules for this node. The Task ID, Party ID and Party Type are supplied by the Input Model at runtime.		
Validation Rules	Each code node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Task ID	String	Yes	This is the Task ID of the current task.
Party ID	String	Yes	The identifier of the user performing the task suspension. This must be the same as the identifier of the user that has the task checked out.
Party Type	String	Yes	This is the system name of the party that Party ID is an instance of.

Comments	String	No	Notes that should be added to the task history. The task history should also be updated to show the date and time that the task was put back as well as the Party ID.
Outcome outputs	**OK**		
None			
Outcome outputs	**Fail**		
None			
Outcome outputs	**NoAccess**		
None			

4.59 Trigger external event

Model node type group	General
Palette	External event nodes
Description	This node component is used to trigger an external event. This means associating some trigger data with the event, changing the status to become **Triggered**, writing a history record to record the change of status and resuming any nodes waiting for this event. If the event is already in the **Triggered** state then no action is taken; it becomes a null operation. Any associated nodes for this event are ignored if they are in the **Complete** or **Superseded** states, so only waiting nodes are resumed. If an external event identifier is provided to the node then only that individual event is processed. There are three valid combinations of parameters when the event identifier is not provided. These are: • EventType, EventSubtype, PartyID and PartyType • EventType, EventSubtype and PartyID • EventType, EventSubtype and ContractID The use of these combinations may result in more than one external event being processed. Each matching event is processed in the same manner as for an individual event as described above.
Label	Text from the configured node description or NLS text for **Trigger external event** if no description is configured.
Verification Rules	None.
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.
Node configuration	
None	
Inputs	

Name	**Type**	**Mandatory**	**Description**
Event ID	String	Conditional (see description above)	Identifier of the external event. If this parameter is supplied, all the other look-up values are ignored.
Event type	String	Conditional (see description above)	Identifier of the event type (for example Receipt of document).
Event subtype	String	Conditional (see description above)	Identifier of the event subtype (for example Application form, Driver’s licence).

Party ID	String	Conditional (see description above)	The identifier of party to whom the event is related.
Party Type	String	Conditional (see description above)	Identifier of the party type to whom the event is related.
Contract ID	String	Conditional (see description above)	The identifier of the contract to which the event is related.
Trigger data	Data Object	No	This data object will be used to supply data about the event. The data object will vary according to the event type. The node component is simply responsible for passing it through to the waiting node (or database).
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Triggered rows	Integer	N/A	This is the count of the number of matching rows found by the look-up process.
Outcome outputs	Not expected		
None			
Outcome outputs	Fail		
None			

4.60 Update task

Model node type group — Workflow nodes

Palette — Workflow nodes

Description — This is a multi-function node component used to reschedule, re-route or re-prioritise a task by changing one or more of its:

- Target execution date
- Task owner (that is the party who should execute the task, where the party may be a team)
- Priority level

This node component will typically be used in conjunction with the **Update Task** custom interaction.

Tasks are required to be checked out prior to this operation and will be put back as a consequence.

Label — Locale sensitive version of **Update Task Properties**

Verification Rules — There are no verification rules for this node. The Task ID, Party ID and Party Type are supplied by the Input Model at runtime.

Validation Rules — Each code node must have 1 arc entering and >= 1 arcs leaving.

Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
Task ID	String	Yes	This is the identifier of the task to update.
Party ID	String	Yes	The identifier of the user performing the update. This must be the same as the identifier of the user that has the task checked out.
Party Type	String	Yes	This is the system name of the party that Party ID is an instance of.
Priority	Integer	No	The new task priority to be assigned.

Activation date	Date/Time	No	The new date and time for when this task will be activated.
Routed to Party ID	String	No	The identifier of the new owner of the task. (Must be used with Party Type). This input has been deprecated – use AdditionalRoutedToParties instead.
Additional Routed To Parties	Data Object Collection	No	A collection of Routing entries data objects that define the parties that the task has been explicitly routed to.
Description	String		
Execution Windows	Data Object Collection	No	A collection of Execution Window data objects that define the time slots for the task to be completed. Once the execution windows have expired, the task is no longer served up through GetWork, but can be executed using PickWork.
Deadline	Data object	No	A Deadline data object that defines when the task must be completed by. The **data/time** deadline attribute specifies an actual deadline for the task, and takes precedence over any Deadline Period that may have been specified. A **deadline period** attribute can be specified that is used to calculate the deadline date/time from the point in time at which the task is activated. The deadline period value is the number of minutes until the deadline. A deadline period is ignored if a Deadline date/time is specified. The **Deadline Warning** attribute specifies the number of minutes prior to the deadline at which the task is escalated to urgent status so that it takes precedence over all other non-deadline tasks. If the Deadline Warning value is not specified, the task’s escalation interval is used instead. The Deadline warning value is used to calculate a Warning date time that is saved in the task instance record.
System name of Party type	String	No	This is the system name of the party that **Routed to Party ID** is an instance of. (Must be used with Party ID).
Comments	String	No	Notes that should be added to the task history. The task history should also be updated to show the date and time that the task was put back as well as the Party ID.
KeepCheckedOut	Boolean	No	Flag to indicate whether the task should be kept checked out or not. The default value is ‘False’, so if a value is not specified the task will be checked in.
Outcome outputs	**OK**		
None			
Outcome outputs	**Fail**		
None			
Outcome outputs	**NoAccess**		
None			

4.61 Wait for external event

Model node type group	General
Palette	External event nodes

Description	This node component is responsible for creating an association between this Node instance and an external event. The Model and Node GUID values for this Node instance are provided to the database so that it can record this association when the external event has not yet been triggered. If the external event is in the **Created** state then the provided GUID values are associated with this event and this new association set into the **Created** state. This node then becomes suspended until such time as a trigger node causes the node instance to be resumed, at which time the node completes with an outcome of **OK** and returns any trigger data that has been set for the event. A history row is written to record the act of creating a new association with this event. If the external event is in the **Triggered** state then there is no need to create a new association as the node can complete immediately. In this case any trigger data defined for this event is returned as part of the node outcome of **OK**. A history row is written to record the fact that an already triggered event was triggered again. If this node is superseded then any association between this node instance and the external event is changed into the **Superseded** state. This ensures that any future trigger node does not cause this node to be resumed. A history row is written to record the change in state of the association. This type of node should not normally be used in any model that is run synchronously from an Operation, since it is an error for such a model to suspend. However, if a value is passed to the input **TimeoutSeconds** then the node stays in memory for this length of time without suspending, so in this case alone it may be used in an Operation. One possible usage is in combination with the Create Task node: the Create Task node can be configured to trigger an external event when the task has been created, and an Operation may be configured to wait with a timeout for the external event before opening the newly-created task. However, be aware that this consumes resources on the CRM server for the duration of the wait, so be sure a) to configure a reasonably short timeout and b) to handle the case of a timeout occurring. If a value of zero is passed as **TimeoutSeconds**, the node will instantaneously check to see if the external event has already been triggered, returning the **OK** outcome if it has and the **TimedOut** outcome if it hasn't.		
Label	Text from the configured node description or NLS text for **Wait for external event** if no description is configured.		
Verification Rules	None.		
Validation Rules	Each node must have 1 arc entering and >= 1 arcs leaving.		
Node configuration			
None			
Inputs			
Name	**Type**	**Mandatory**	**Description**
EventID	String	Yes	This is the identifier of the event row which the node must update with the Model and Node GUIDs.
TimeoutSeconds	Integer	No	If this input is set then the node will wait for at most this number of seconds before returning the outcome **TimedOut** if the event has not been triggered. This value must be greater than or equal to zero. See notes above for further details.
Outcome outputs	**OK**		
Name	**Type**	**Mandatory**	**Description**
Trigger data	Data Object	N/A	A dynamically constructed data object containing information supplied by the **Trigger external event** node. If the event has already been triggered, this data will come from the TRIGGER_DATA column in the event row.
Outcome outputs	**Fail**		
None			
Outcome outputs	**TimedOut**		
None			

4.62 Permit model execution

Model node type group	Security nodes
Palette	Security nodes
Description	This node is used to authorise a user’s session for executing restricted models.
Label	None.
Verification Rules	None.
Validation Rules	This node must have 1 arc entering and 1 arc leaving it.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A
None	

4.63 Deny model execution

Model node type group	Security nodes
Palette	Security nodes
Description	This node is used to remove authorisation for executing restricted models in a user’s session.
Label	None.
Verification Rules	None.
Validation Rules	This node must have 1 arc entering and 1 arc leaving it.
Node configuration	
None	
Inputs	
None	
Outcome outputs	N/A
None	