Docflow Documentation

Docflow

The Docflow module is a subsystem for automating business processes. In fact, any transaction, order, production cycle, etc., are reflected in a chain of related documents. Thus, the process model can be mapped to a set of documents: contracts, invoices, payrolls, acts of acceptance, etc. These documents are executed sequentially or in parallel.

The source data, tasks to perform, intermediate and final results are contained in a set of interrelated documents. The execution of documents is entrusted to the participants. The document changes from one state to another after the participant completes the current task.

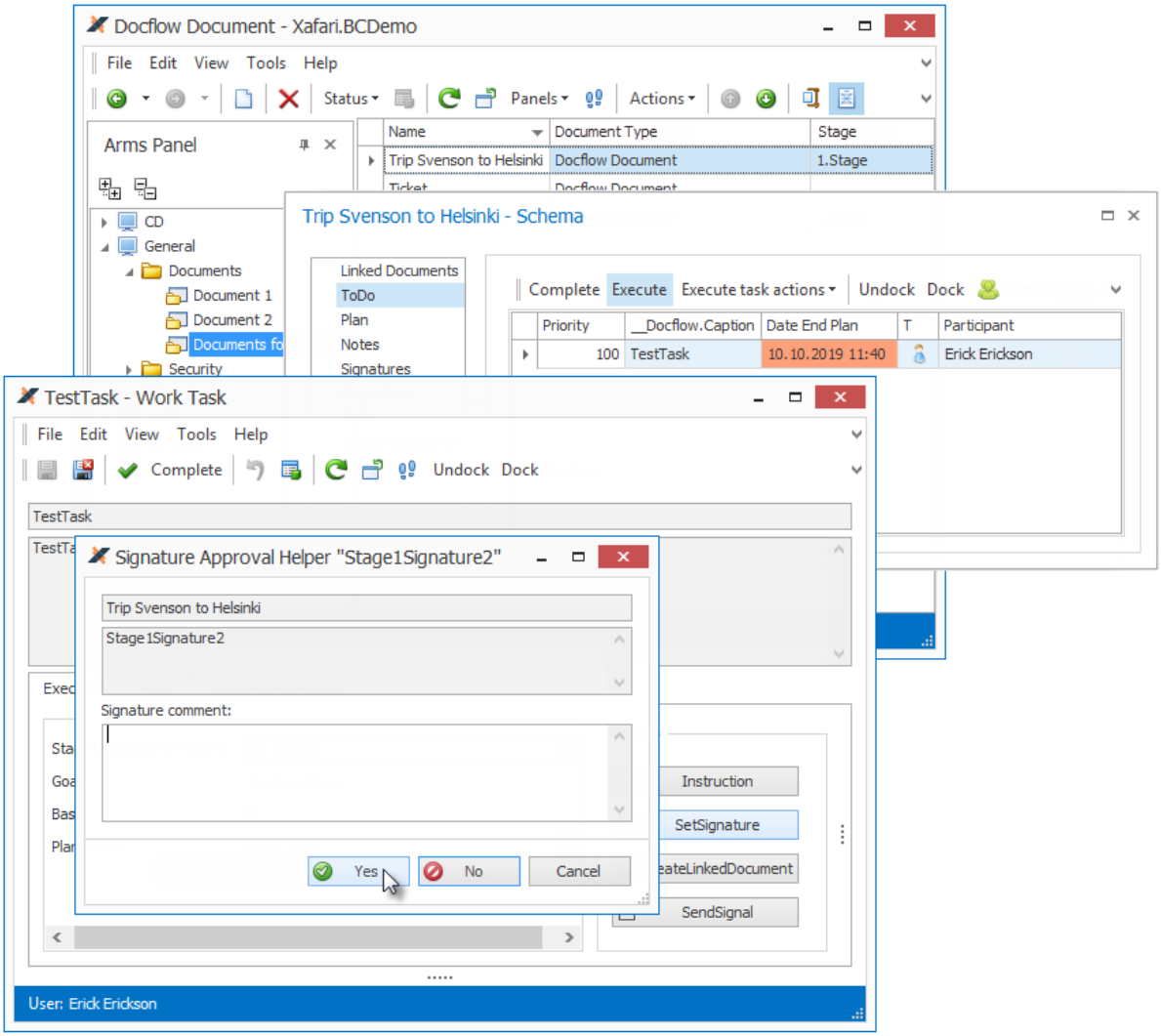
Docflow Document is an object that has special features, this object is dynamically binded to a Schema, which formally describes its life cycle. It is the Schema that determines the behavior and properties of the Document in accordance with the model of a particular business process. After creating a Document, the Docflow service monitors and manages the events and states of this object.

The Schema describes the sequence of stages, through which the Document goes, and the Tasks to be performed. The Schema is developed in design mode, using the Model Editor, also, future versions of Xafari Docflow will support XAS. Customization of the Scheme is available even when the system is used in production. It is possible to develop both rigid fixed Schemes like Workflow, and flexible ones in the style of Adaptive Case Management.

The concept of the Docflow module can be compared with the State Diagram or the Finite State Machine. The system that includes the Docflow module acquires the qualities of robustness, it is able to adapt to the new requirements of a changing environment.

To get started with Docflow, it is recommended to read the HOW TO CREATE A BUSINESS APPLICATION USING XAFARI AND XAF post. Along with the basic Xafari development techniques, it explains how to integrate Docflow Document into the data model and configure Participants, and also describes the design and execution of a simple Schema.

To see a variety of Schemas and use cases, refer to the **General|Documents** section in the **Xafari BC Demo** installed with Xafari.


Docflow Document - Xafari.BCDemo
File Edit View Tools Help
Status Panels Actions
Arms Panel
Name
Document Type
Stage
Trip Svenson to Helsinki
Docflow Document
1.Stage
CD
General
Documents
Document 1
Document 2
Security
Trip Svenson to Helsinki - Schema
Linked Documents
ToDo
Plan
Notes
Signatures
Complete
Execute
Execute task actions
Undock
Dock
Priority
__Docflow.Caption
Date End Plan
T
Participant
100
TestTask
10.10.2019 11:40
Erick Erickson
TestTask - Work Task
File Edit View Tools Help
Complete
Undock
Dock
TestTask
Signature Approval Helper "Stage1Signature2"
Trip Svenson to Helsinki
Stage1Signature2
Signature comment:
Yes
No
Cancel
Instruction
SetSignature
SendSignal
User: Erick Erickson

Docflow in Application

Introduction

The Document is a business object that dynamically attached to a certain Schema, and its life cycle and behavior are completely determined by this Schema.

An example of extending a Business Model's class with Docflow features is described in the HOW TO CREATE A BUSINESS APPLICATION USING XAFARI AND XAF post, similar manipulations are performed in the source code of the **BC Demo** application. The Schemas available for the Document object are determined in the Application Model, this is explained in the Docflow Composition topic.

This topic describes the Document object in the application, its interaction with the end users, and the Views and Actions that will be generated based on the Schema description.

Docflow Users

There are two categories of objects that can manage the Document state and change it, these are Participants and Curators.

Participants are those who are entrusted to execute Schema Tasks, and this can be either an employee or a system, i.e. the Task will be completed automatically when the assigned conditions are met or events occur. The main usage scenario assumes that Participants are users and roles from the Security System tables. However Docflow provides for the ability to assign Participants from other sources. Configuring Participants is described in the **Participants** section of the Document Structure topic and **Docflow Participant Extension** section of the HOW TO CREATE A BUSINESS APPLICATION USING XAFARI AND XAF post.

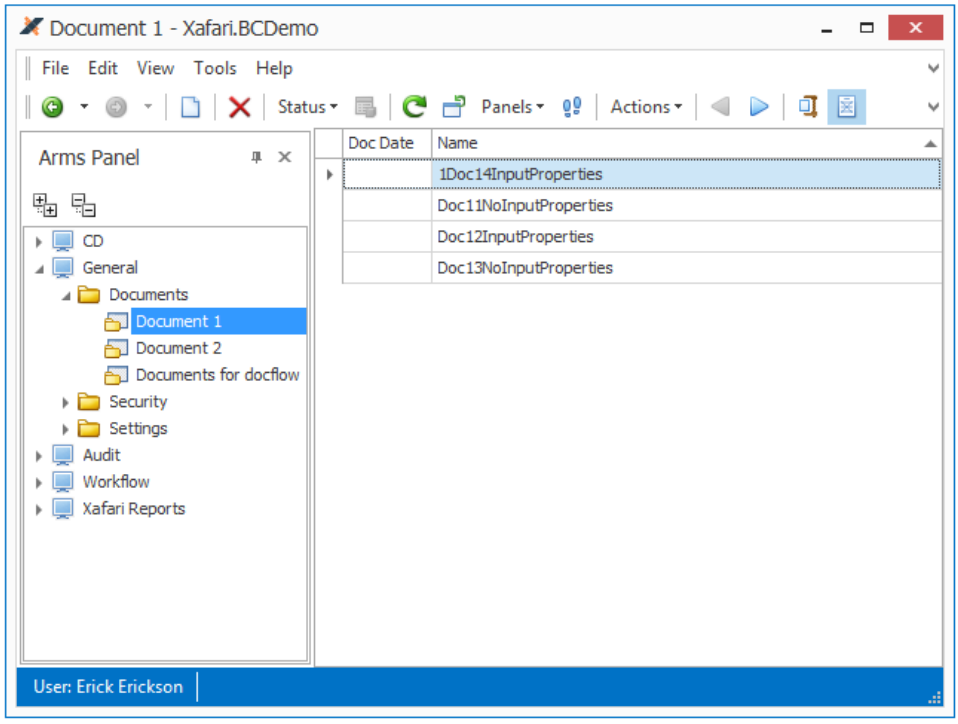
During the processing of the Document, the Participant receives appointments, sees assigned Tasks in the ToDo list, and executes them using specialized Views. This is described below.

Curators oversee the flow and, if necessary, intervene explicitly in the process. The Curator object refers to the Participant object, in fact, it is the Participant with additional **Notification** and **Administration** options. The **Notification** option means that this Curator only receives notifications of significant events of the Document. If the **Admin** option is checked, then the Curator can pause and cancel the Schema, execute any Tasks, reassign the Tasks executors, approve Signatures directly, add new Curators, etc.

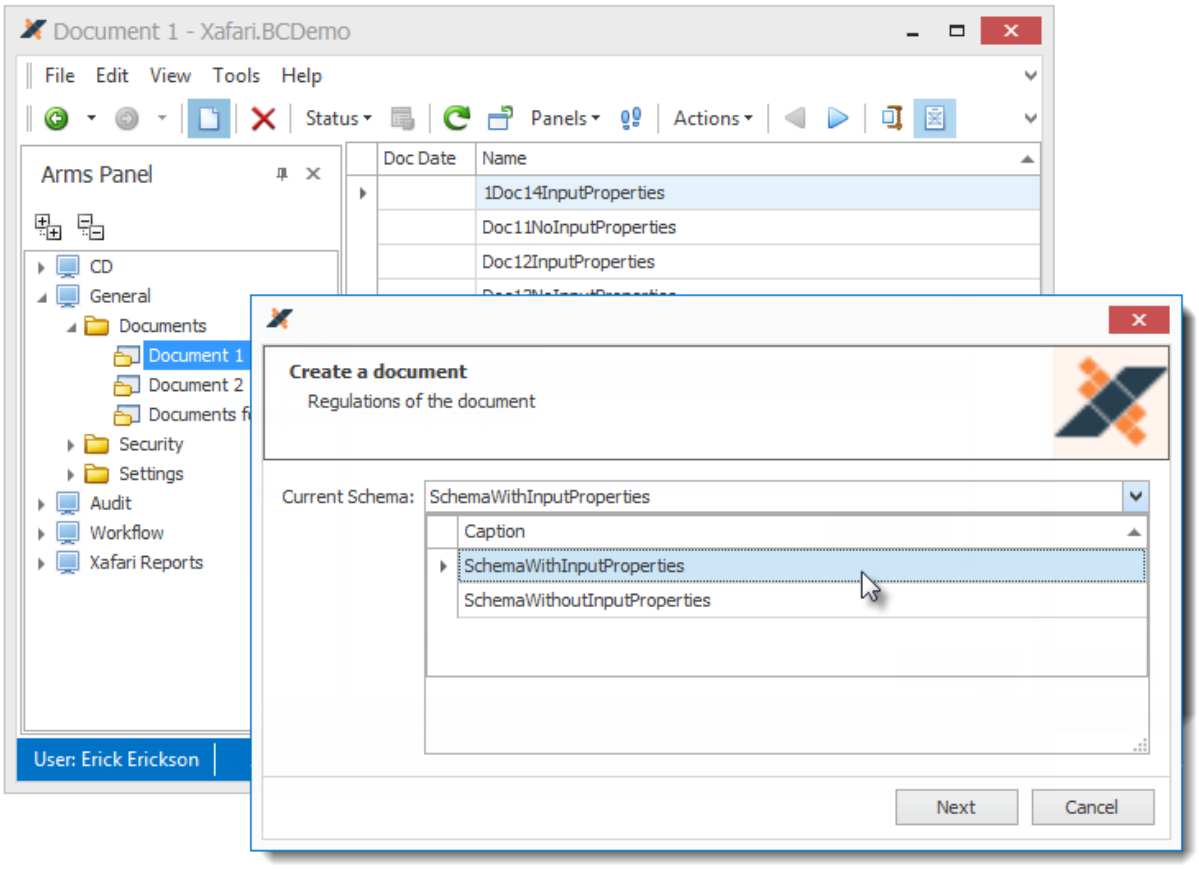
Initial Curators can be defined in the Document class using the **Xafari.Docflow.Curators.CuratorsAttribute** attribute. Further, the Curator can add or remove other Curators, as well as perform various operations, it is described below.

General Operations with Document

Since any Document is a business object of the system, Document Views are opened in the application through the appropriate Navigation Items. In the image below, the various types of Documents are represented in the navigation bar, whose elements provide access to the corresponding List Views.



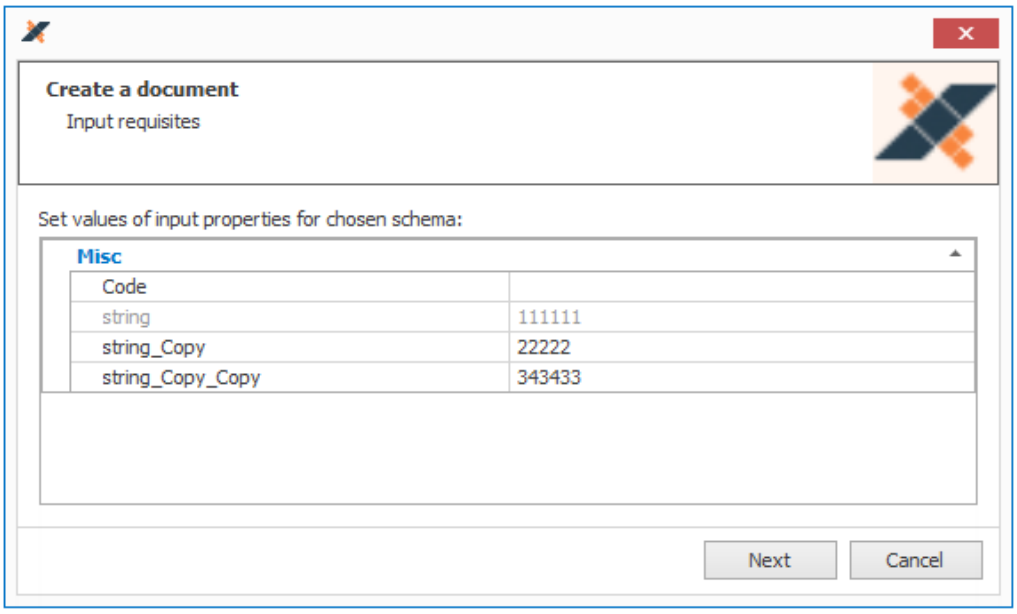
To create a new Document, invoke Document's List or Detail View and execute the **New** Action, usually it starts a Wizard.



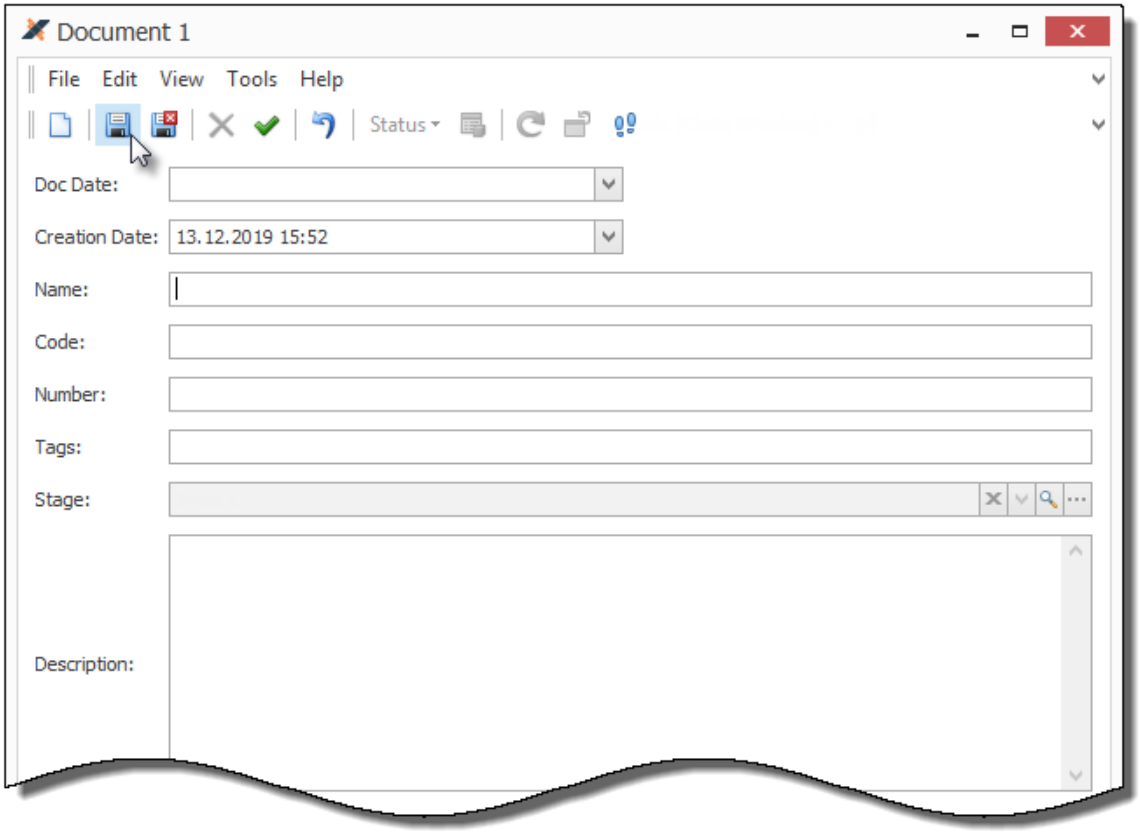
The Wizard steps are determined by how the Document and its Schemas are configured. See Docflow Composition topic.

If the Document has more than one Schema available, the Wizard displays a window to select the current Schema to continue. The above image shows such a case.

Next, if the selected Schema has Properties that are marked with the **IsInput** flag, the Wizard prompts you to set the start values. These are initial values to start a business process.

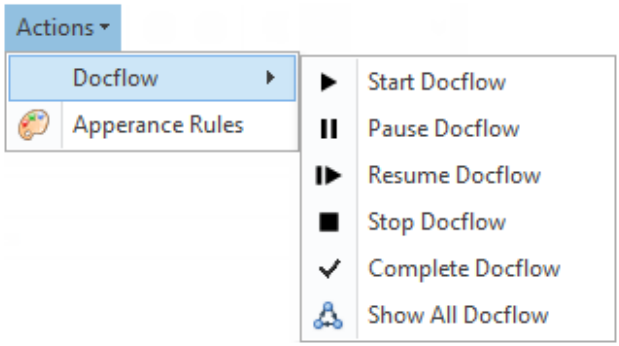


After the Wizard finishes, the new Document's Detail View will be displayed. This is the standard form generated for the persistent type from which the Document was created. It displays the properties declared in this type. At this stage, it is necessary to confirm all the data entered by saving the Document.



At the time of creating the Document object, the Docflow service will generate Signatures, Properties, etc., and then the Schema starts. Further processing of the Document in the context of Docflow will be done via the Schema Panel, this is described in the next section.

The Document Curator also has the ability to globally manage the execution of the Scheme through a special group of Actions.

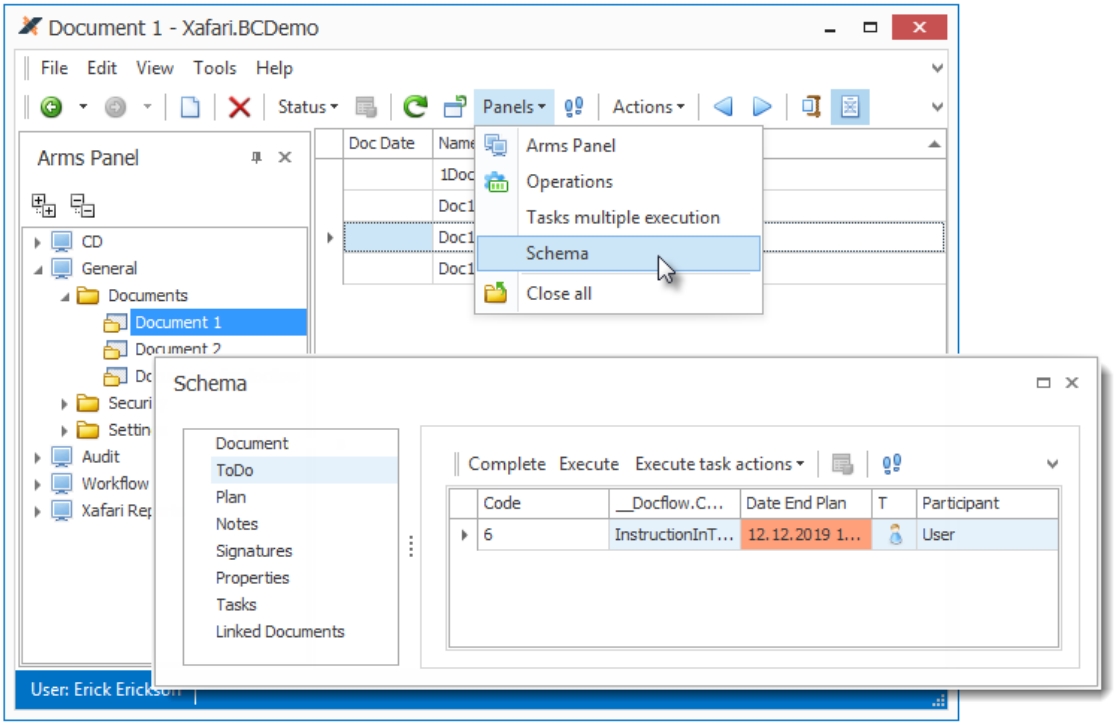


Shema

The structure of the Schema and its constituent elements is described in the Docflow Design chapter. In runtime, Documents are processed using the Schema Panel and the Task List, that provide monitoring of the current state, assigning participants, task execution, logging, etc.

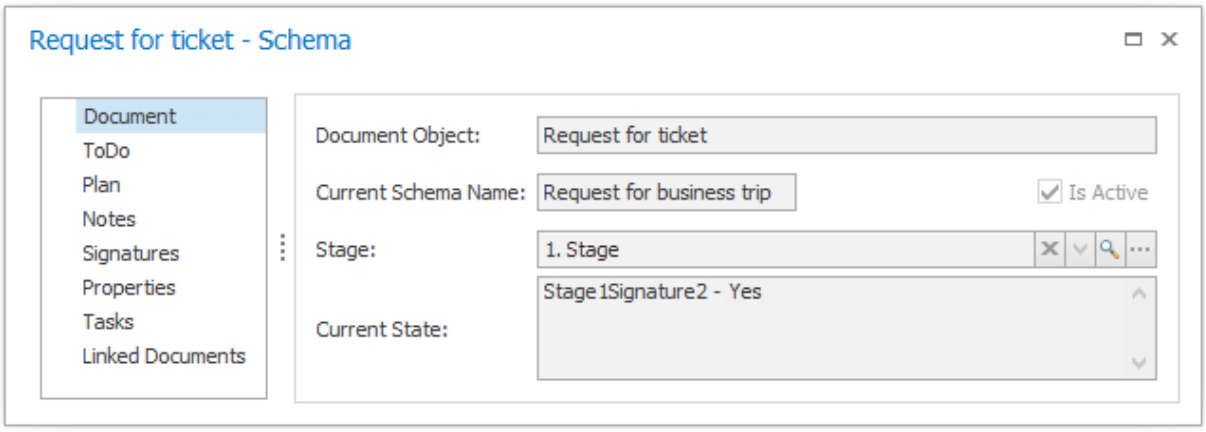
Schema Panel is available via the **Schema** Action from the toolbar or context menu, this panel displays various aspects of processing the current Document, as a rule, it is intended for the Curator.

Task List displays Tasks and provides tools for their management and execution. This is explained below in the Task Management and Execution section.

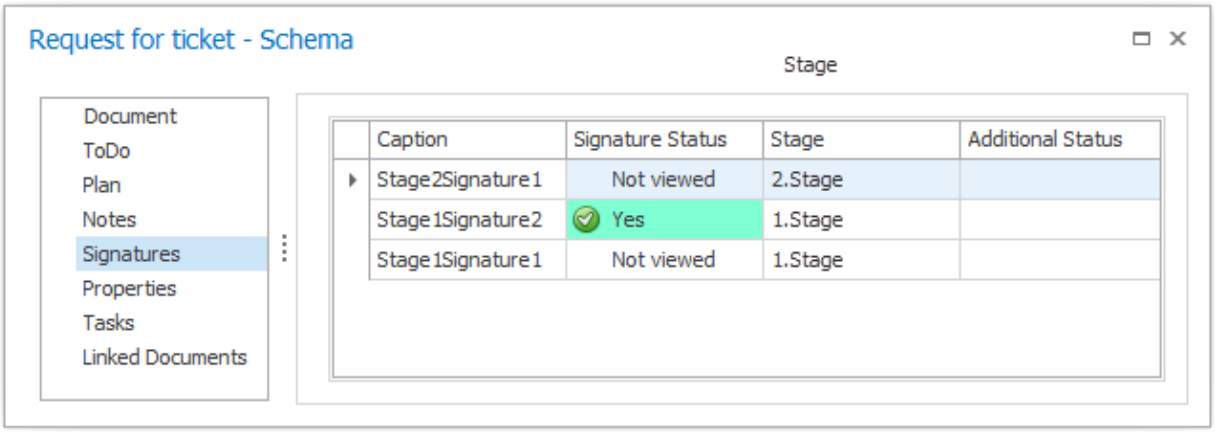


The Shema Panel control can be positioned in a suitable location and its size can be adjusted, its left side contains a list of tabs, and the main area displays the selected one.

- The first tab provides general information about the Document.



- **ToDo** tab displays **DocflowData_ToDoList_ListView** which contains the all active Tasks of the current Document, this is one of the varieties of the Task List.The toolbar of this View contains Actions that allow you to manage Tasks and, in particular, to execute them. Task List and Task execution are described in detail below in a separate section.
- **Plan** tab is a list of Stages, it shows their expected and actual timing. The actual start of the current Stage is the start of the Scheme or the end of the previous One. The Stage is completed when the last of the included Tasks is completed. The beginning of the Stage is fixed once, but the end can be overwritten, this will happen if Docflow return to the previous Stages.
- **Notes** tab shows comments collection, Participants can add comments when executing Tasks.
- **Signatures** list allows to evaluate the status of Document Signatures.

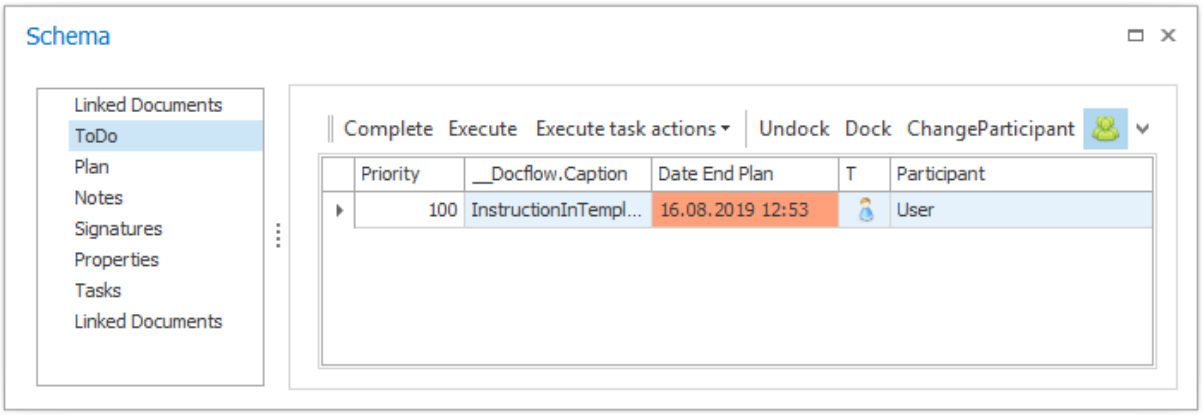


- **Properties** shows current values of Document Properties.
- **Tasks** tab displays a log with information about the execution of the Tasks.
- **Linked Documents** tab contains links to related Documents that were created during the execution of the Schema.

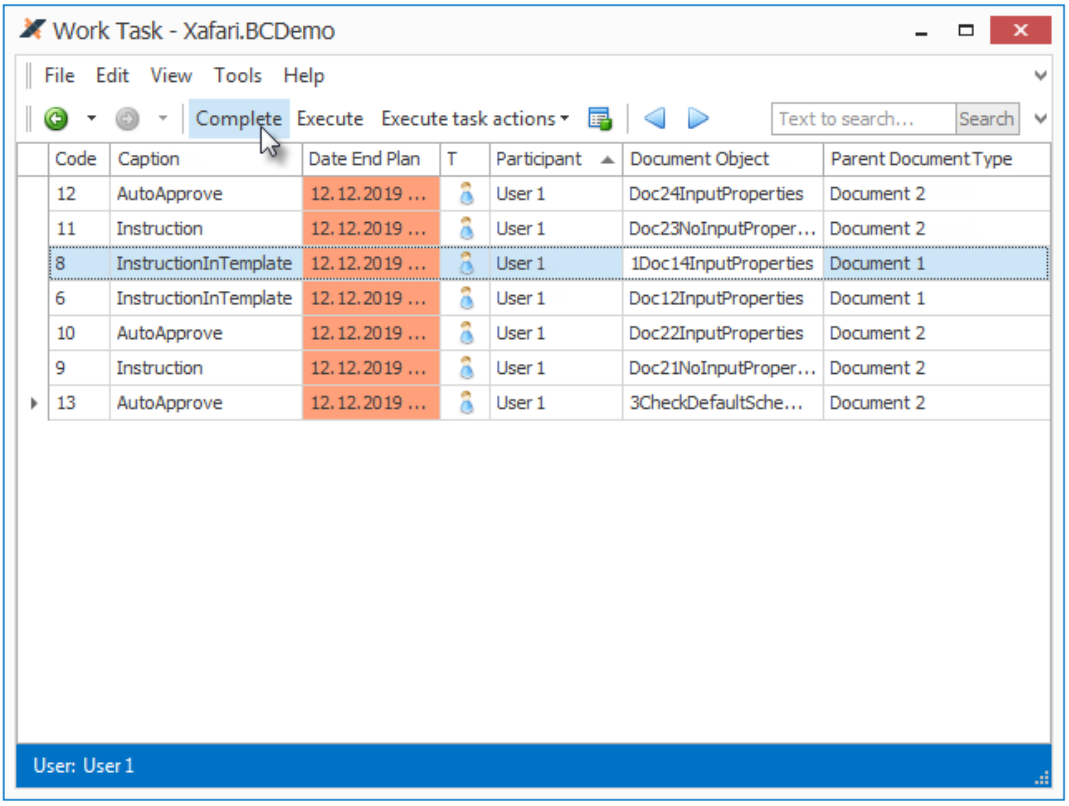
Task Management and Execution

The Task contains Actions to be executed, View Properties, Notes, processing dates, criteria and switches that allow to direct the flow through different branches. Tasks can be combined into a group, which allows to configure some parameters at the group level. Grouped Tasks can be performed sequentially or in parallel, depending on the specifics of the process. Begining Tasks are generated when a Document is saved, further Tasks are generated as the previous ones are completed. The Task can be assigned to both the Participant and the system.

The Tasks are presented in the Task List. As described above, the Schema Panel contains a list of active Tasks in the current Document. This variety of Task List is most needed by the Curator to monitor changes that occur during the execution of Tasks in the selected Document.



If the current user is an ordinary Participant, it is appropriate for them to access all their active Tasks from various Documents, such an opportunity is provided by the **WorkTask_ListView_Docflow**.

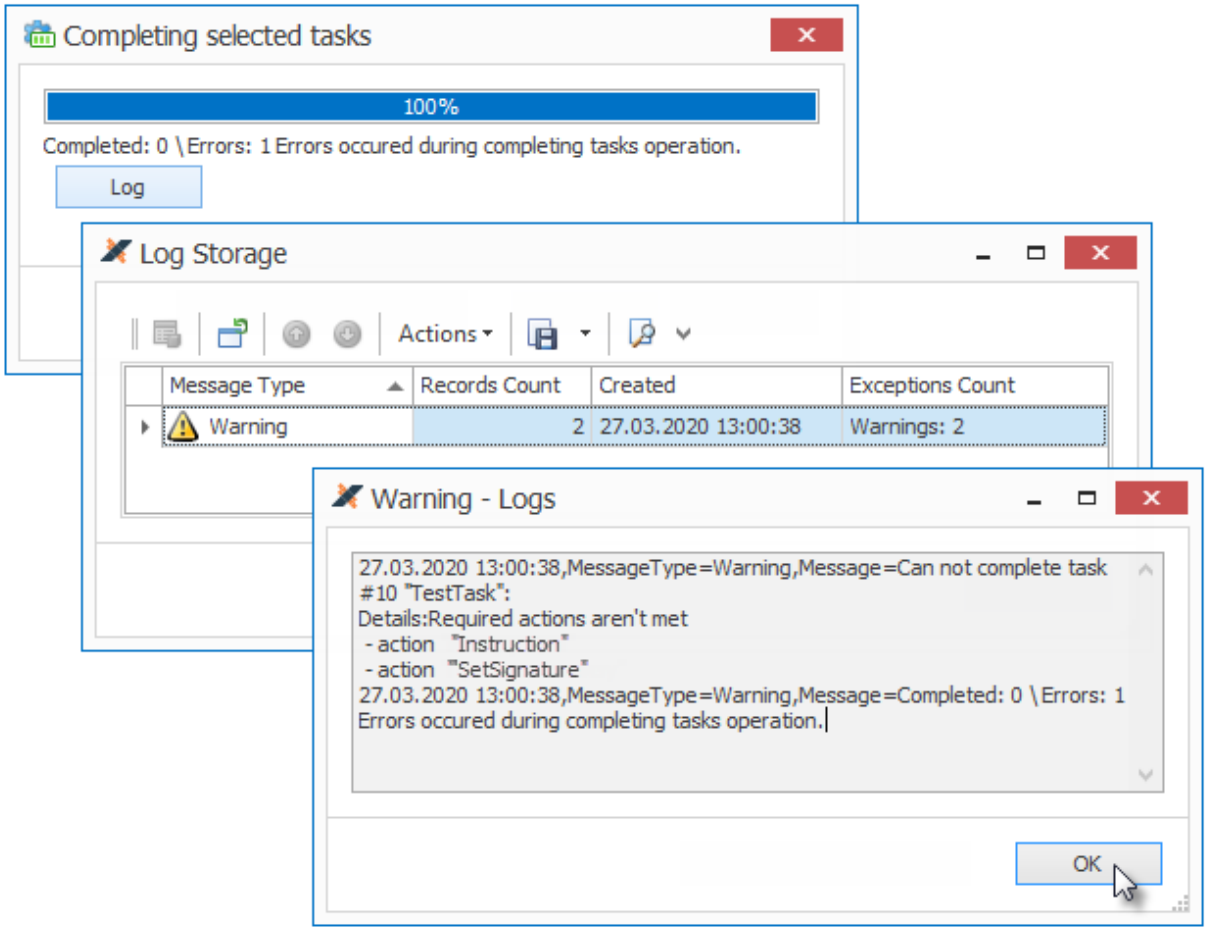


Note

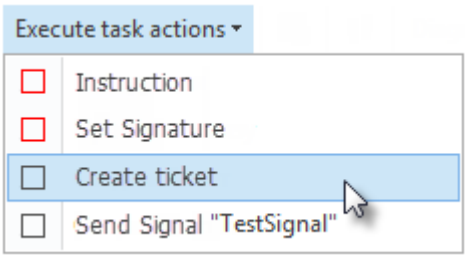
Xafari.BC.Tasks module provides a whole set of Task Lists, among them there are **WorkTask_ListView**, **WorkTask_ListView_Docflow**, **WorkTask_ListView_Docflow_NotActual** and so on. See them in the Application Model and focus on the **Criteria** property. In general, Task List can manage Tasks of arbitrary origin. This documentation considers Tasks generated by the Docflow service.

The Task List's toolbar provides a set of Actions for managing and executing Tasks. The availability of certain Actions is determined by the status of the current user in the Docflow context.

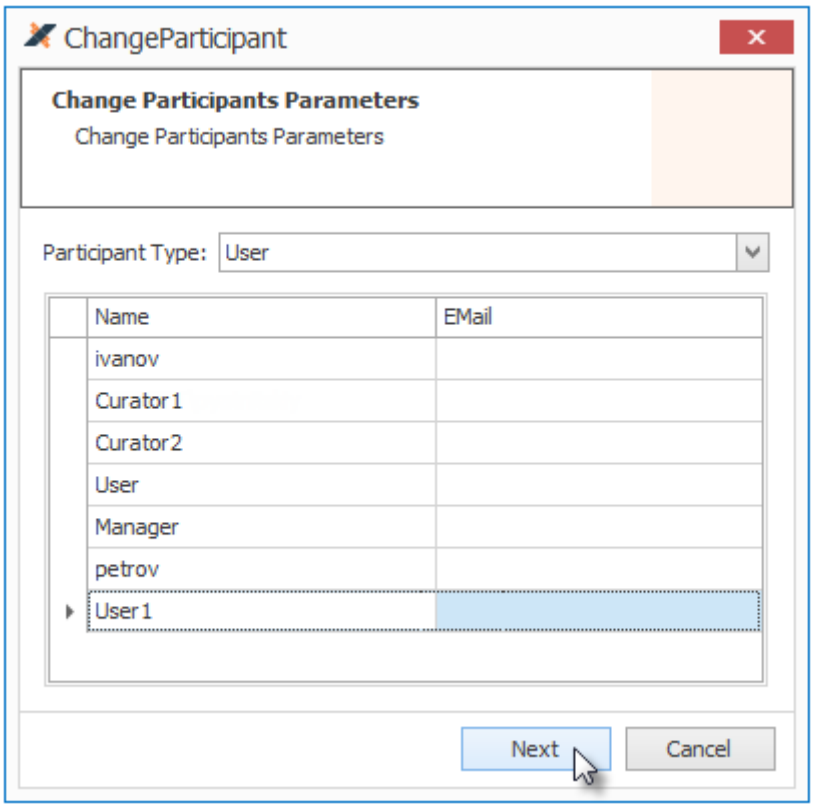
- **Complete**. Even if all the nested Actions are executed, the Task must be completed in order to change the Document state. The Task can be completed immediately if there are no required Actions and the Failure Criterias are not met. When trying to complete the Task, the framework runs a visualized validation operation. If there are no errors, a success message is displayed, the Task is marked as completed, and the next one in the queue is generated. If there are reasons for not completing the Task, the operation displays the corresponding message and writes errors to the Log



- **Execute**. The Task execution assumes the execution of nested Actions and, if necessary, editing the View Properties. First, the several predefined operations will occur: if the Participant is a role, then the Task will be docked to the current user; the actual start date will be set; if the Task contains a single Action, its execution will start automatically. If the Task contains multiple Actions, the user will be directed to the Task Detail View, which provides all the necessary tools, it is described below.
- **Execute task actions**. The user can execute nested Actions separately.

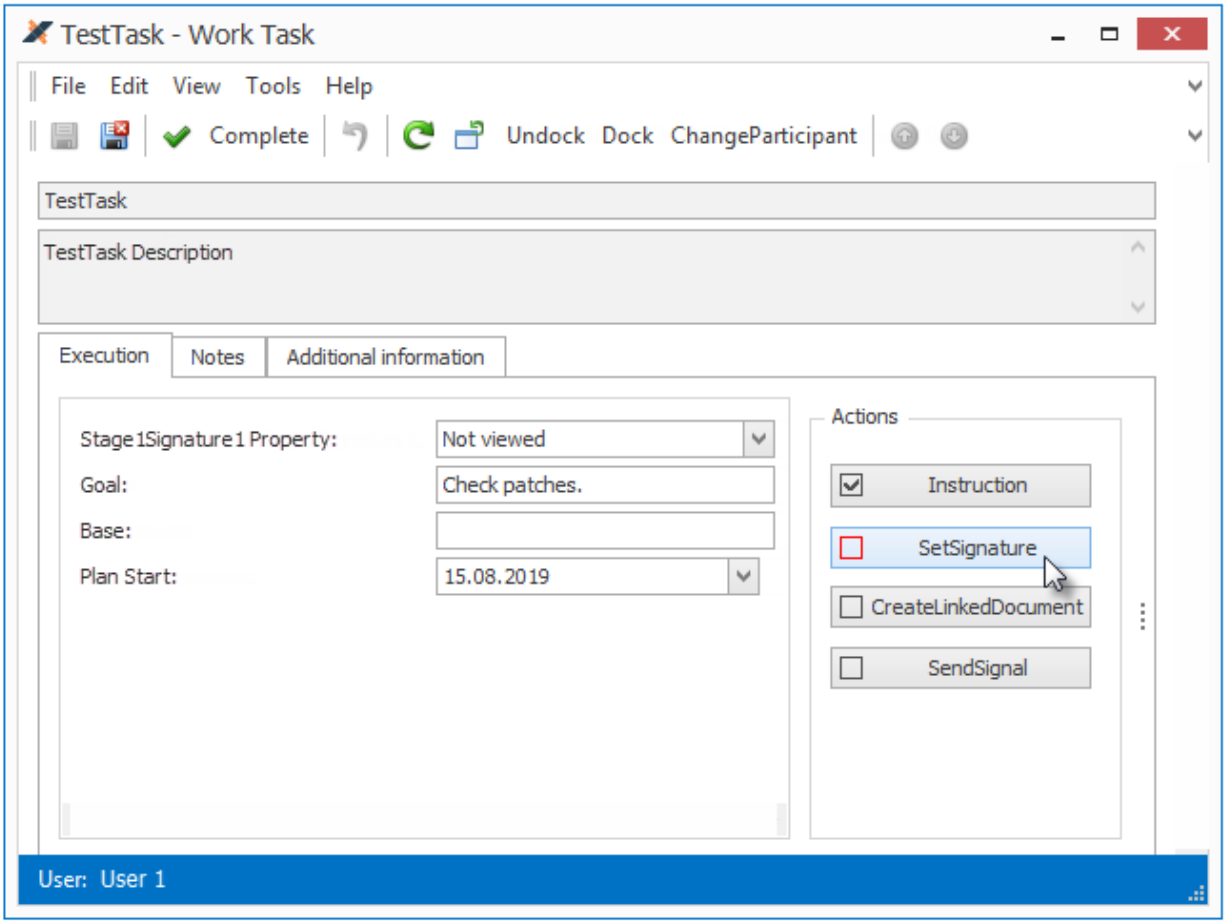


- **Undock**. Resets the executor if it was assigned via the **Dock** Action.
- **Dock**. The current user becomes the executor of this Task.
- **ChangeParticipant**. Assign this Task to the specified Participant.



- **Show All Tasks**. Displays all current Tasks of the Document, this is active only for the administrator.

Task Detail View displays View Properties and nested Actions on the **Execution** tab.



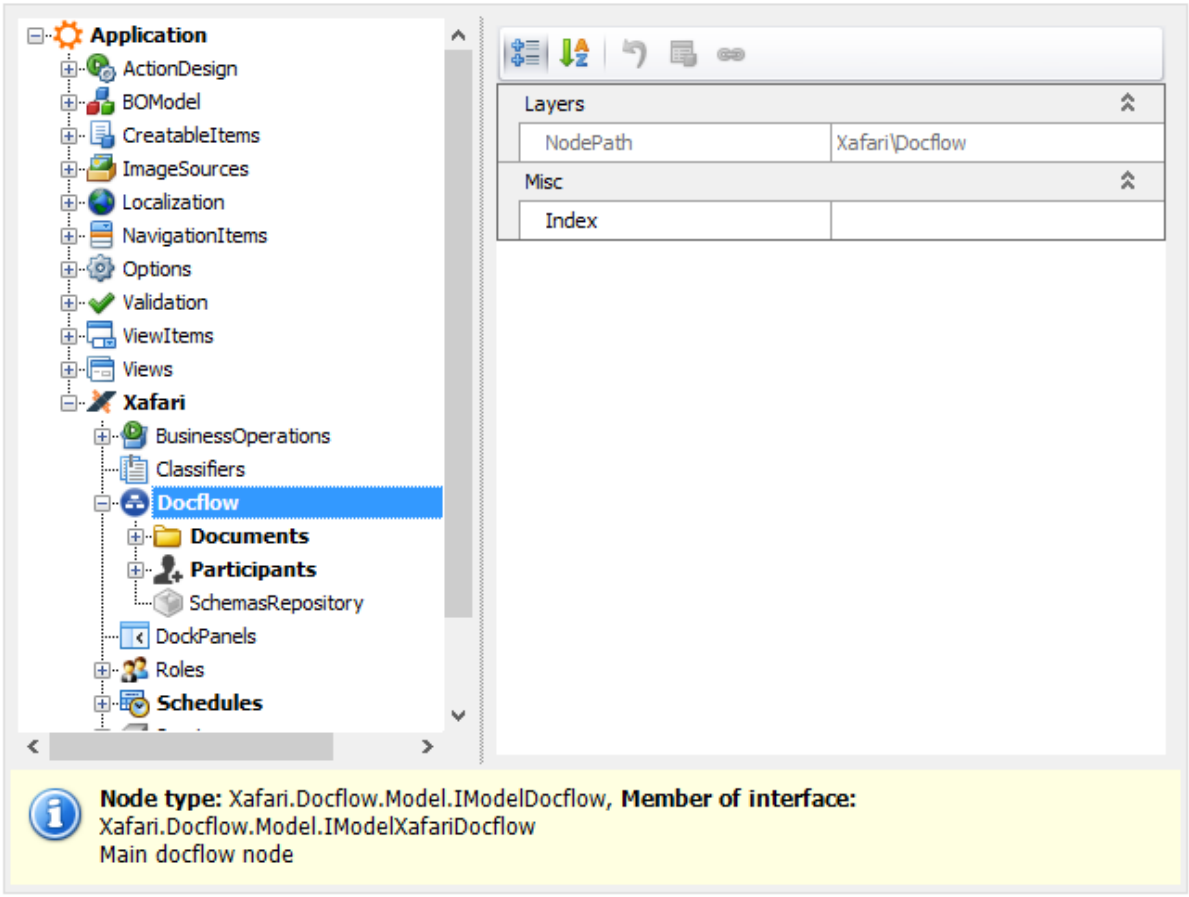
View Propertis is a subset of Shema Properties that are available for editing by the executor when working on the Task. It is possible to adjust the visibility of some properties depending on the values of others. Actions provide access to specialized operations, for example, send a Signal, approve a Signature, etc. The required Actions are marked with a red square, the Task cannot be completed until they are not executed. Executed Actions are ticked off.

Docflow Design

Topics in this chapter provide information about configuring the Docflow subsystem.

Before proceeding, take a moment to review the HOW TO CREATE A BUSINESS APPLICATION USING XAFARI AND XAF post. It examines the general concepts of Xafari and the integration of the Docflow service in a business application. It is also necessary to have a detailed look at the Documents presented in the **BC Demo** application installed with Xafari.

Documents and Schemes are designed by the developer via the Model Editor. Future versions of Xafari plan to support Docflow in XAS. The Docflow settings are presented as a complex tree-like structure, the top of which is the **Xafari|Docflow** node.



The **Documents** node is intended for the actual modeling of business processes, it collects Documents definition and Schemes for their processing. Other nodes are auxiliary, the elements contained in them can be used in the design of Schemes.

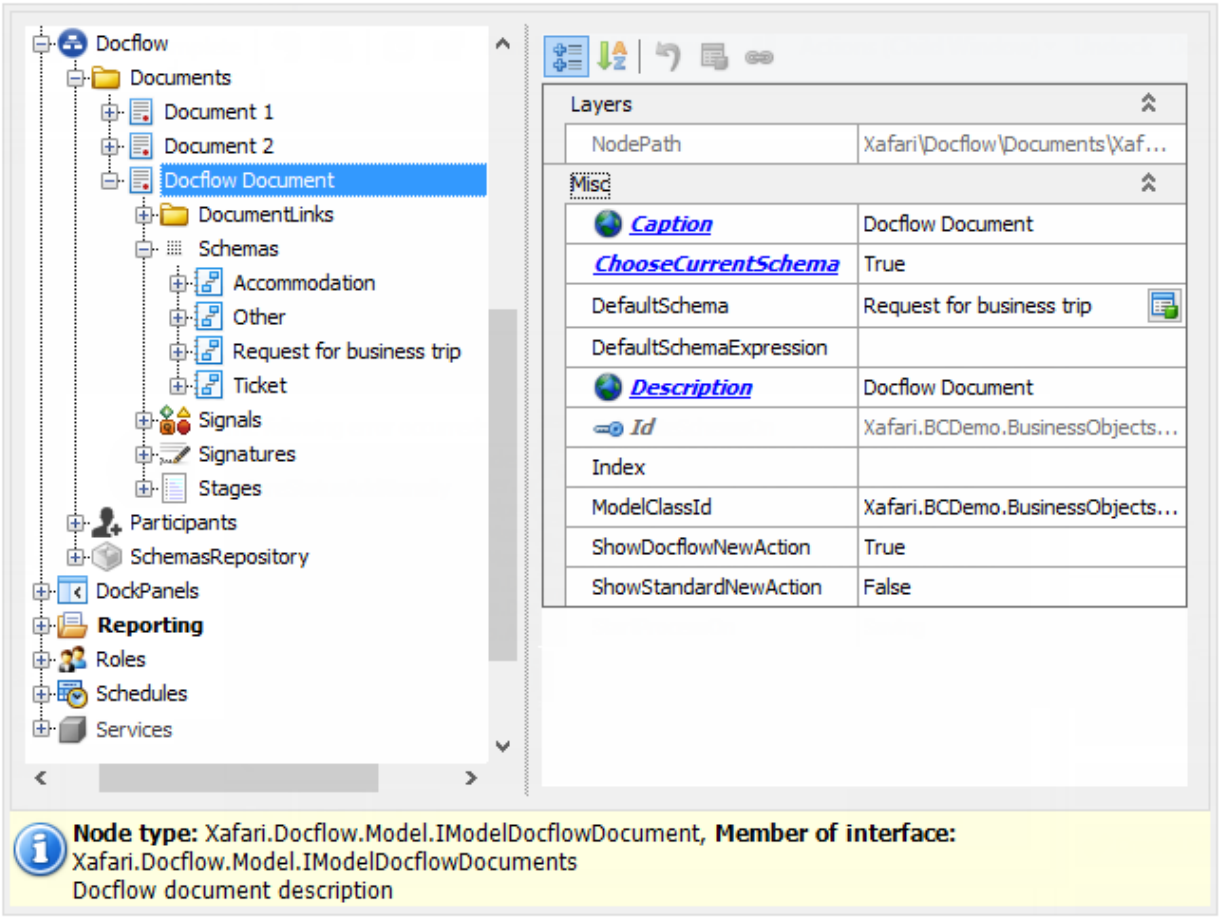
Document Structure

Docflow options are divided into three main groups:

- Documents
- Participants
- Schemas Repository

Documents

Document item binds a persistent class to specific features, the structure of this item and its settings determine the behavior of the object in the application. The same type cannot be repeated in different Documents.



Properties:

Property	Description
ModelClassId	Indicates a persistent type that implements the **IDocflowSupport** interface.
ChooseCurrentSchema	Document may have several associated Schemas. This property specifies whether or not to choose the current Schema when creating an object (a Document instance). **True** value indicates that an

	additional window with a list of available Schemas will be displayed.
DefaultSchema	Specifies the Schema to use by default.
DefaultSchemaExpression	Contains an expression that evaluates the default Schema. The priority of this property is higher than the **DefaultSchema**.
ShowDocflowNewAction	This boolean property determines the availability of a specific Docflow **New** Action.
ShowStandardNewAction	This boolean property determines the availability of a standard XAF **New** Action.

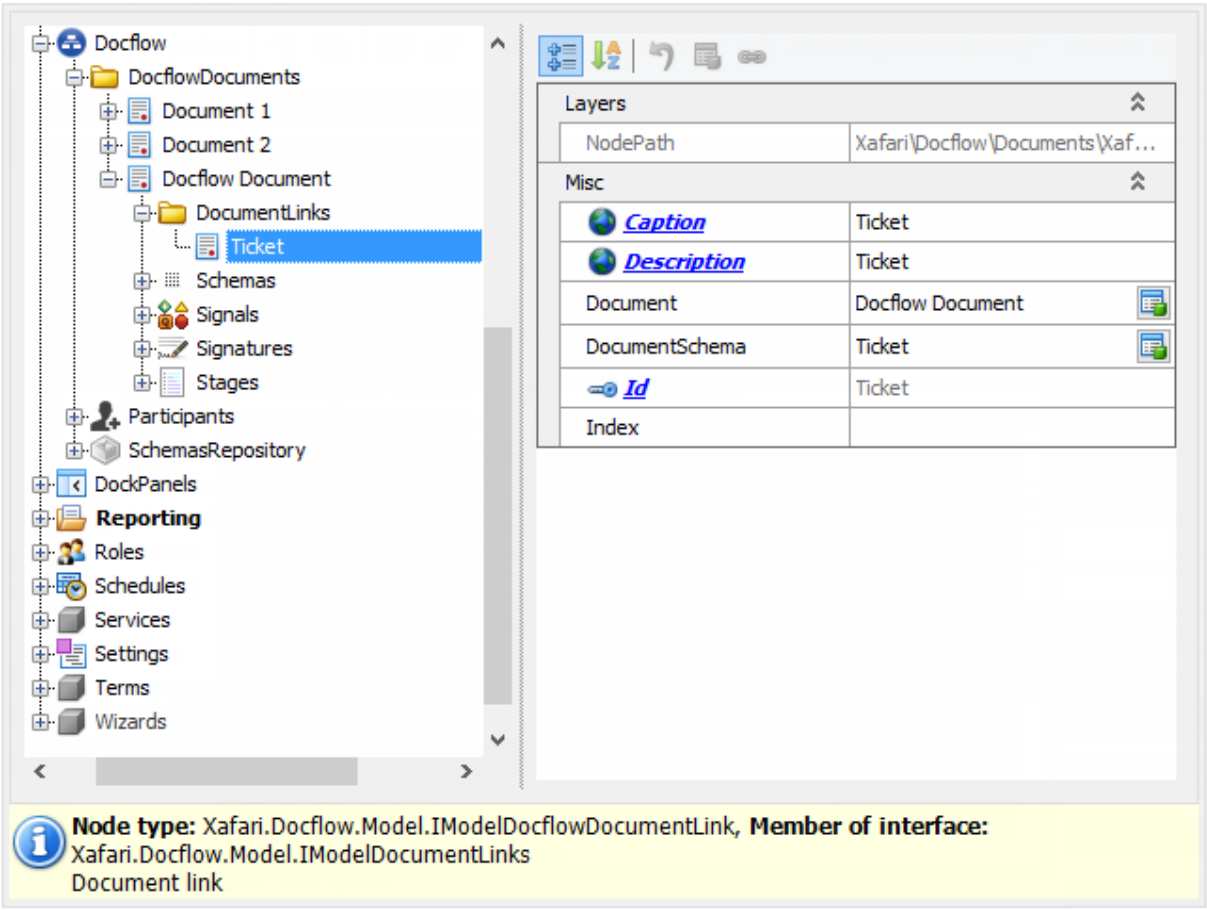
Collections:

Collection	Description
DocumentLinks	Links to related Documents.
Shemas	The Schema models the production path through which the Document must pass. One Document can have several Schemas to choose from.
Signals	The Signal allows to coordinate multiple related Documents. Each Shema can generate or respond to Signals.
Signatures	The Signature is a specialized checkpoint that simulates an approval mark on a Document. Approval allows the Document to move forward, and rejection-returns it for revision.
Stages	Stages allow to divide many Tasks into segments and get a Document plan.

The Shemas is described in a separate topic. Signals, Stages and other nodes collect auxiliary items that will be referred in Schemas, these nodes are explained in the following sections.

Related Documents

DocumentLinks node lists dependent and parent Documents. Schemas of related Documents can interact with each other by means of Signals. The Schema of the current Document can contain a Task that creates a subordinate Document. A similar node is also present in the Task Template, if such a Template is added to the Scheme then the Documents mentioned in It will also be taken into account.

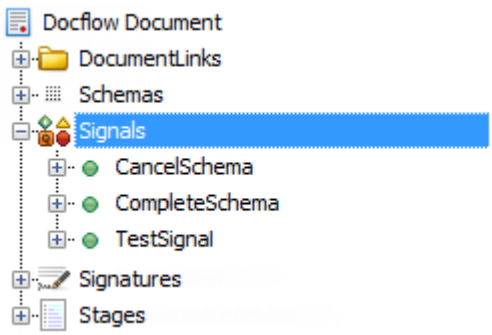


Properties:

Property	Description
Document	Specifies the type of linked Document.
DocumentSchema	Specifies the Schema of the linked Document.

Signals

Schemas can send and wait for Signals that are defined in this node. Sending a Signal occurs when certain conditions are met, and receiving a Signal initiates the execution of subsequent Tasks. The Schema manages Signals using special **Send signal** and **Wait signal** Actions. **Signals** node is also present in the Task Template.

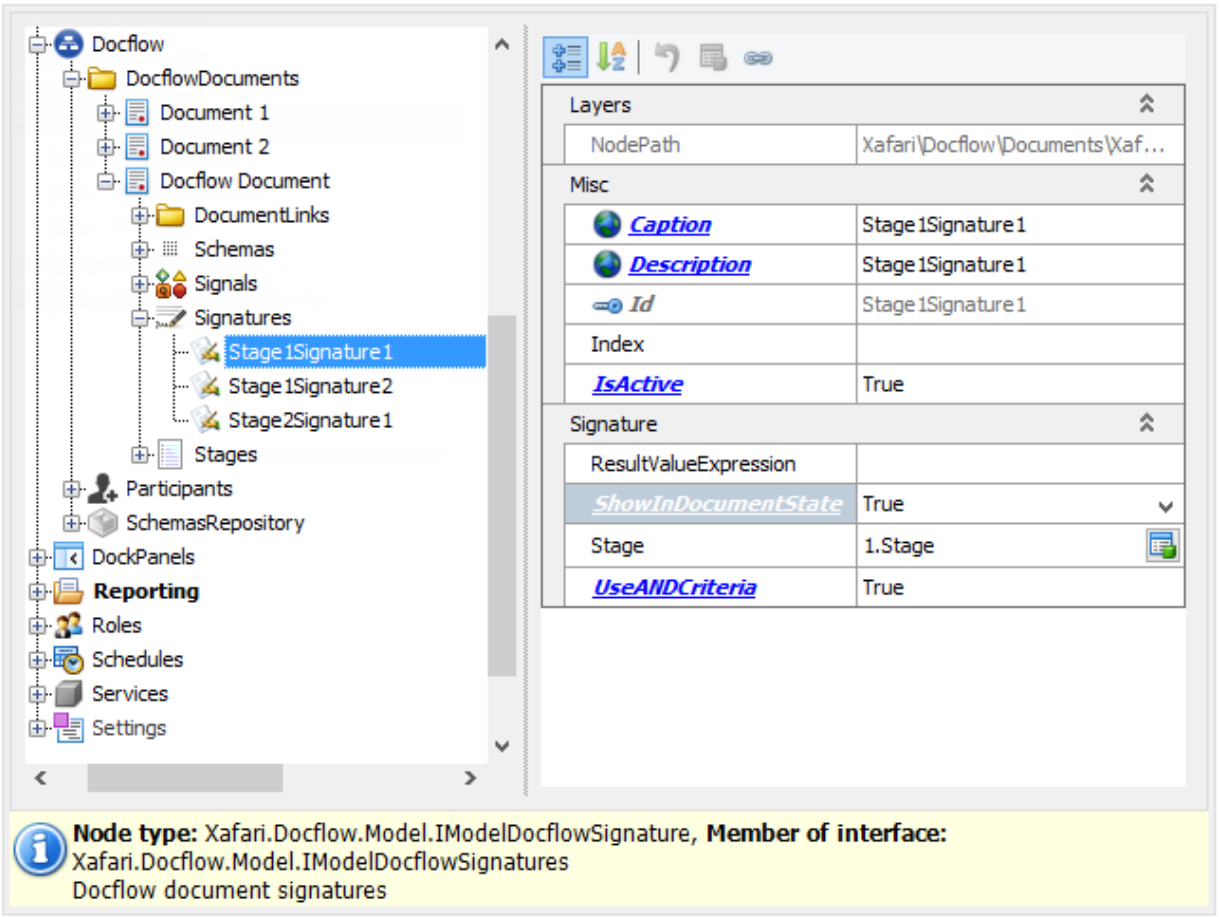


Document interaction with Signals is provided by two Actions: Send Signal and Wait Signal. In Tasks, Signals can affect Failure Criteria and Switches.

Signatures

Signature simulates the fact of verification of the Document by the responsible person or automatic operation. The status of the Signature can be **Yes** or **No**. A value of **No** requires a mandatory comment. It is possible to reuse the same Signature by different Executors in different Tasks. This is because the Signature status is accumulated in the collection of key-value pairs, and the final result is calculated using logical operations AND or OR. For example, a production plan must be approved by multiple managers. You can define the corresponding Tasks and use the same Signature in them. Then, the plan can be considered approved if at least one Executor made an affirmative decision, or if everyone did so.

In Schemas, Signatures are managed by a special Docflow Action.



Properties:

Property	Description
Stage	Linking a Signature to one of the available Stages. Sighting, as a rule, completes a certain Stage of Document.
ShowInDocumentState	The flag indicates whether the Signature is displayed in the current state of the Document.

UseANDCriteria	Specifies the logical operation that will be applied to the collection of values to get the final result. If this property is set to **True**, the system will apply the AND operator, otherwise it will be OR.
ResultValueExpression	Specifies an Expression to calculate the value of the Signature. If the expression is specified, the evaluated value takes precedence over the contents of the collection.

Stages

A Stage is an organizational unit that aggregates a subset of Tasks. An ordered sequence of Stages forms the plan of the Document. Stages facilitate the perception of a long and complex Schema and simplify process management. Overcoming a particular Stage can affect the status of the Document. The Document is considered completed only after all the Tasks that are included in the plan are completed.

Properties:

Property	Description
Deadline	The expression that sets a deadline.
Signal	The Signal that will be sent when the deadline comes.
SetStatusEntry	Specifies the status that will be set for the Document at the start of the Stage.
SetStatusExit	Specifies the status that will be set for the Document when the Stage is completed.

SendSignalParametersMapping collection contains parameters for the Signal used.

Participants

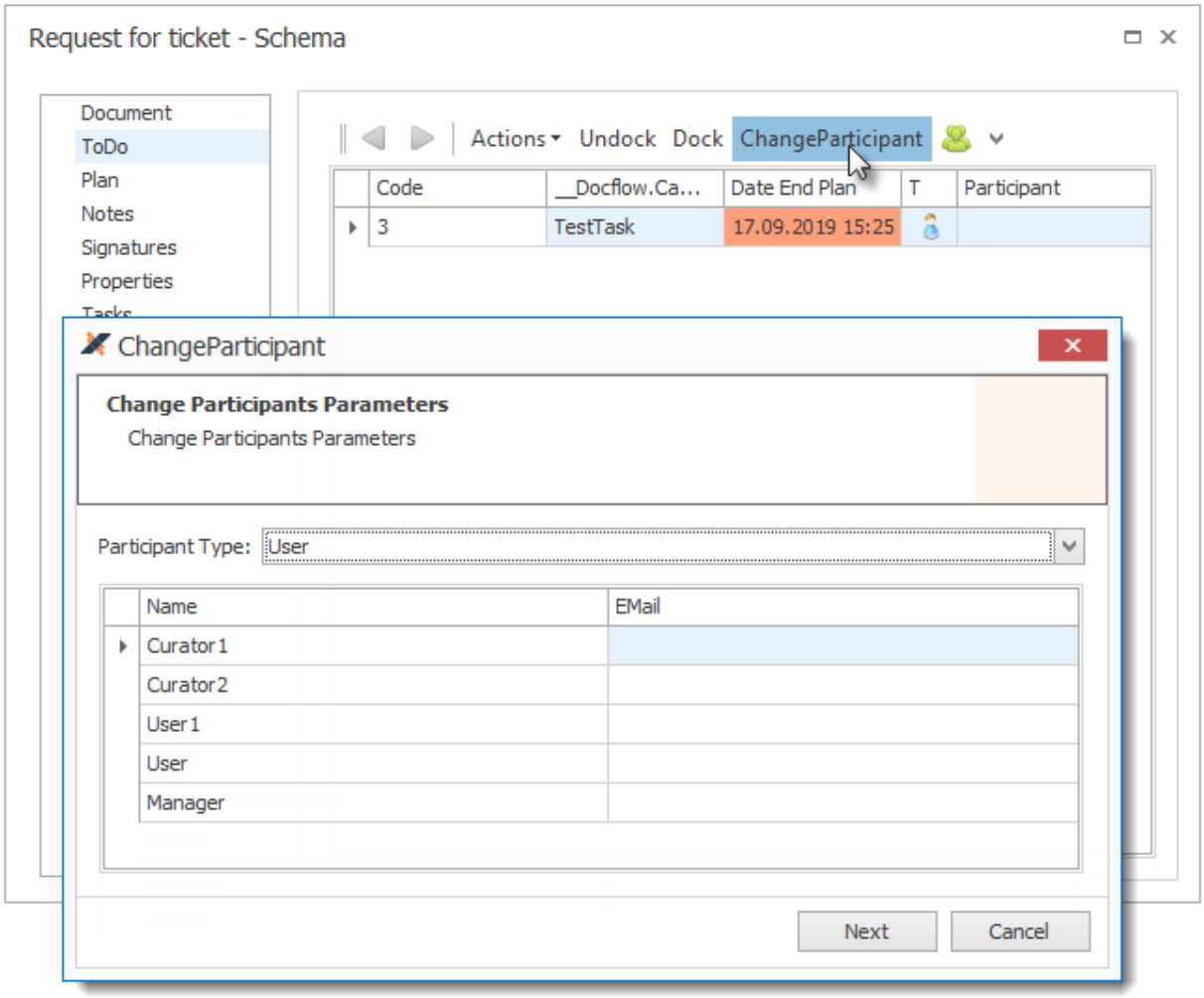
The **Participants** node lists objects that represent some of the persons responsible for performing Tasks. It is important to note that this node contains previously known Partisipants, i.e. the corresponding objects must be guaranteed to exist in the runtime. By default, Participant providers are **Task_TaskUserParticipant** and **Task_TaskParticipantRole** tables (**Xafari.BC.Tasks.DC.TaskUserParticipant** and **Xafari.BC.Tasks.DC.TaskParticipantRole** types correspondingly). Each element in the **Participants** node must match a record in one of these tables, in this case, the Docflow service will set this object as an Executor when generating the Task.

However, the Partisipant provider can be replaced, the type that registers as a new provider must inherit the IUserParticipant interface.Most often, the concept of a Participant is mapped to a system user and a role. This approach is used in the **BCDemo** app supplied with Xafari. You can view the appropriate code in the

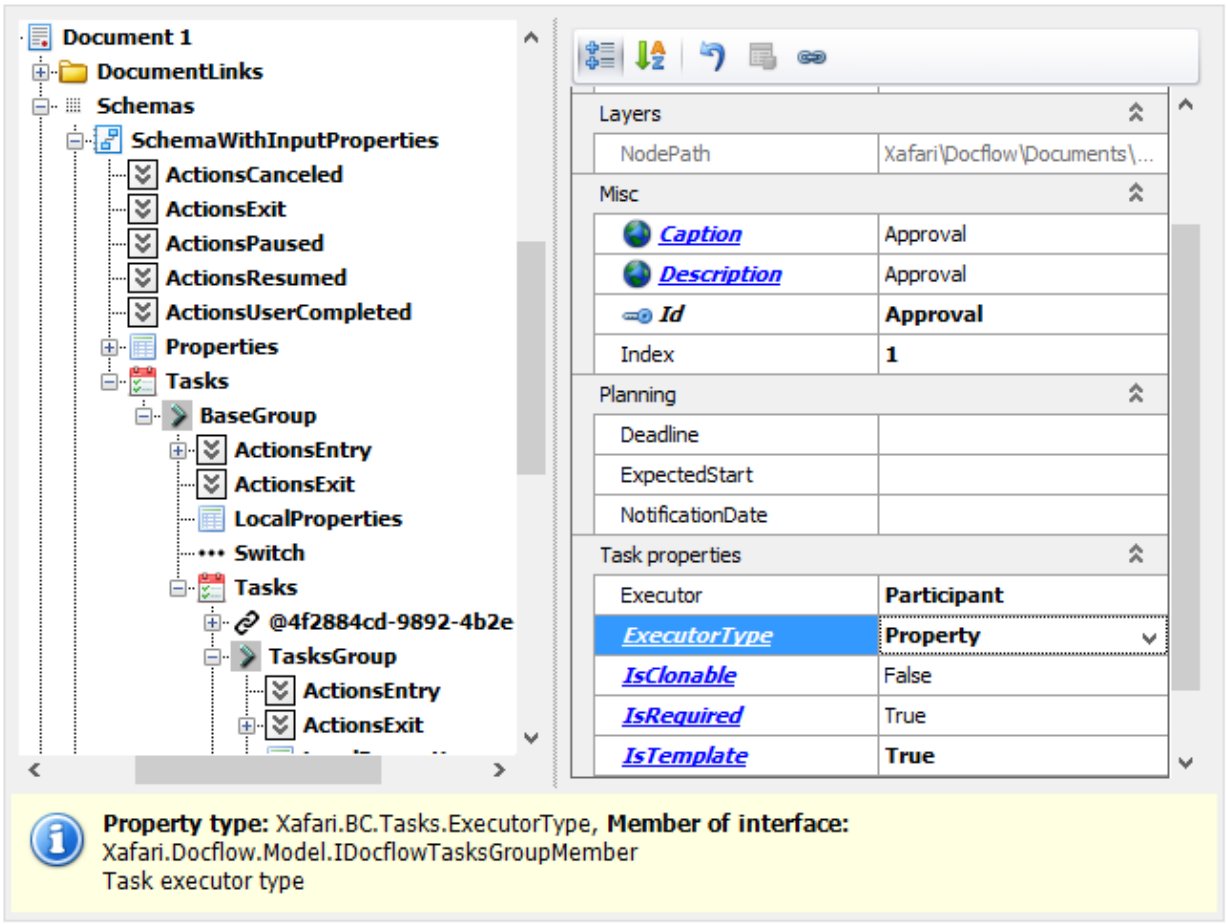
Xafari.BCDemo.BusinessObjects|ParticipantExtensions.cs and *Xafari.BCDemo|BCDemoModule.cs* files.

Partisipants objects from the Application Model are usually used in designing, testing, debugging Schemas, and so on. But this node is not the main means for assigning Tasks to a specific responsible person (or their group). After the system has been deployed and commercial operation has begun, the set of Partisipant objects changes dynamically and can hardly be controlled by the designer of the Schemas and Tasks.

The decision on the appointment of Executors is appropriate to take at the stage of creating the Document object or even later, as the Stages pass and Task pools are formed. This should be done by the user responsible for processing the Document. ToDo list on the Schema Panel provides special Actions that allow to manage the Participants of each Task individually. The image below shows the **Dock**, **Undock** and **ChangeParticipant** Actions, these procedures are described in the ToDo List topic.



An experienced Schema developer can manipulate Task Executors using Schema Properties. Advanced configuration techniques allow to calculate the Executor through Expressions when generating Task, specify it in the Wizard at the stage of creating the Document, etc. To get such an experience, it is necessary to analyze in detail the **SchemaWithInputProperties** example from the **BCDemo** application.

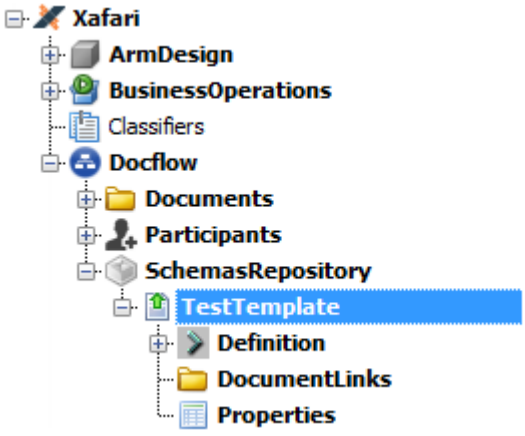


There are three types of Participants:

- User
- Role
- User group

Schemas Repository

The **SchemasRepository** node is intended to store Tasks and Task Groups, that are commonly used in the Schemas. They can be described in the form of templates and added to the Schema as needed. The **SchemaRepositoryItem** contains the **Definition**, **DocumentLinks** and **Properties** collections. The structure and parameters of a **Definition** node are similar to a Task Group node. Properties are described in the corresponding topic. **DocumentLinks** are described in this topic above.



Docflow. Schema

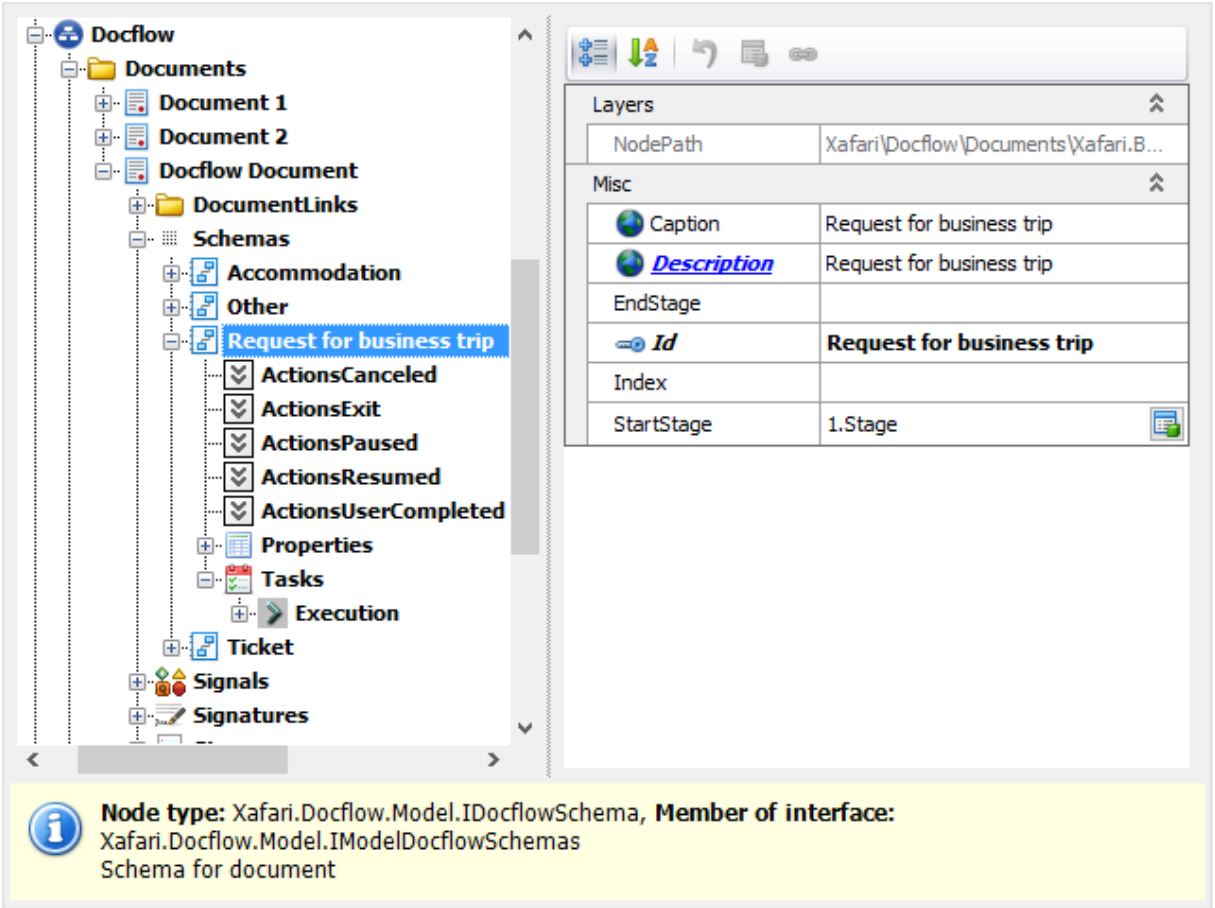
Introduction

Schema is a formal description of processing a Document in runtime. As already noted in the Docflow Composition topic, the Document node attaches a persistent type to Schemas, Signals, Signatures, and other Docflow components. A single Document may contain a collection of Schemas, and the current Schema must be specified or calculated when the Document object is created or saved.

It is the current Schema that gives the final structure and functionality to the Document object. The Schema brings together all the Docflow components, it defines for the DocflowManager service when and how to generate Tasks, calculate a plan, synchronize flows, send Signals, calculate expressions and check conditions, perform switching and much more. This whole set of measures ensures the fulfillment of a business task, fixing all the intermediate and final results.

Before proceeding, it is recommended that you review the **Docflow Introduction** and **Docflow.Schema Design** lessons. This topic explains the Schema node and its nested components.

Description



Properties:

Property	Description
EndStage	The final Stage of the Schema
StartStage	The initial Stage of the Schema

Collections:

- Actions are specific components of the Docflow that provide the execution of predefined atomic operations. The Schema can include Actions that will be performed automatically by the system at the appropriate event: Canceled, Exit, Paused, Resumed and User Completed.
- Properties are additional specifications of the Document, they are generated dynamically when the object is created and extend its structure.
- Tasks collection contains Task Groups and links to Templates from the **SchemasRepository** node.

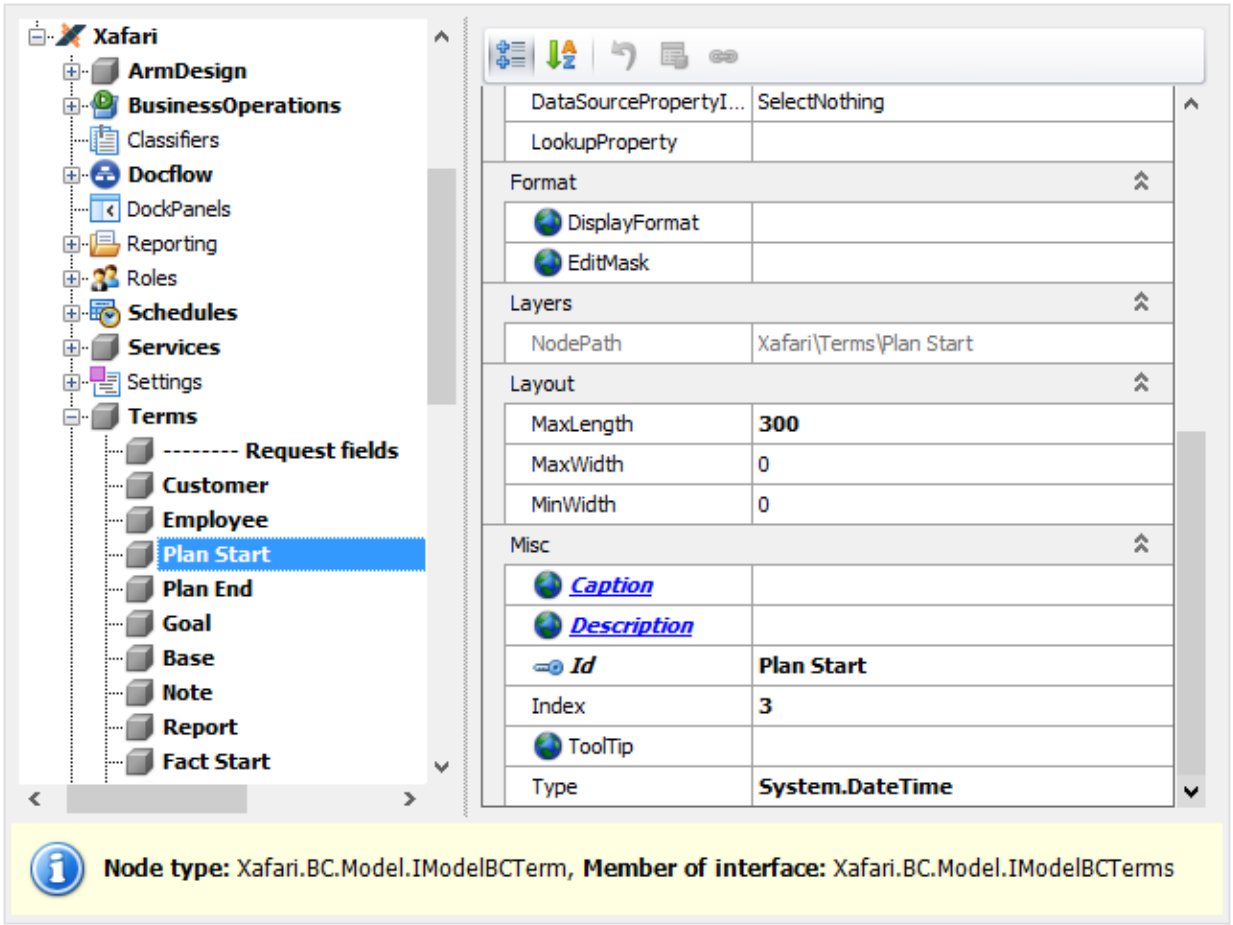
Docflow. Properties

Introduction

Docflow Property is a special kind of variable that is declared at design-time, and not in the code, it is based on the Dynamic Properties technology. Properties allow to extend the Document's structure with any details, regardless of what persistent type is associated with the Document. Since the Property can be defined at the Schema level, thus, the structure of the Document object created in runtime is determined by the current Schema.

Properties are local and global depending on which node they are added to. The Properties of the Schema or SchemaRepositoryItem are global, and the Properties of the Task and Task Group are local. Local Property is a reference to some global Property.

The Property is bound to a specific type by means of a Term, which also sets a number of parameters for editing the value. The Term must be predefined in a special node.

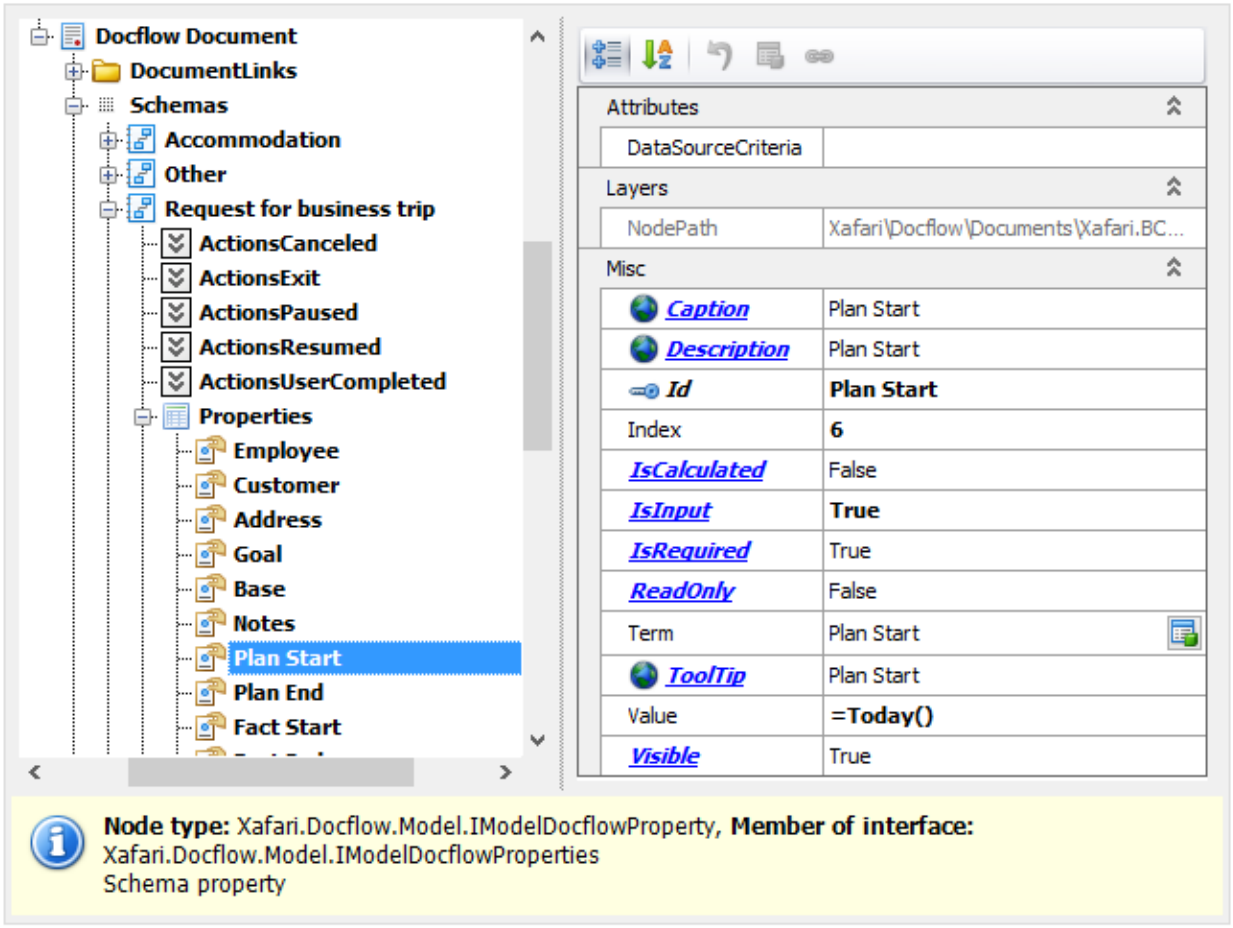


There are four kinds of Properties:

- Property
- Binding Property
- Signature Property
- Collection Property

Property

The simple Property describes the value of the specified type.



Properties:

Property	Description
DataSourceCriteria	Specifies the criteria expression used to filter the referenced objects displayed in a Lookup Property Editor. The value will be interpreted similarly to DataSourceCriteriaAttribute.
IsCalculated	Boolean value indicating whether or not the Property is calculated. The calculated Property uses the expression specified in the **Value** parameter. User input is ignored.
IsInput	Boolean value, if it is **True**, then the user is prompted to input a value when creating a Document.
IsRequired	Boolean value, if it is **True**, then the value must be set. If the **IsInput** and **IsRequired** flags are **True** at the same time, the system will require a value to be entered when creating the object. Another

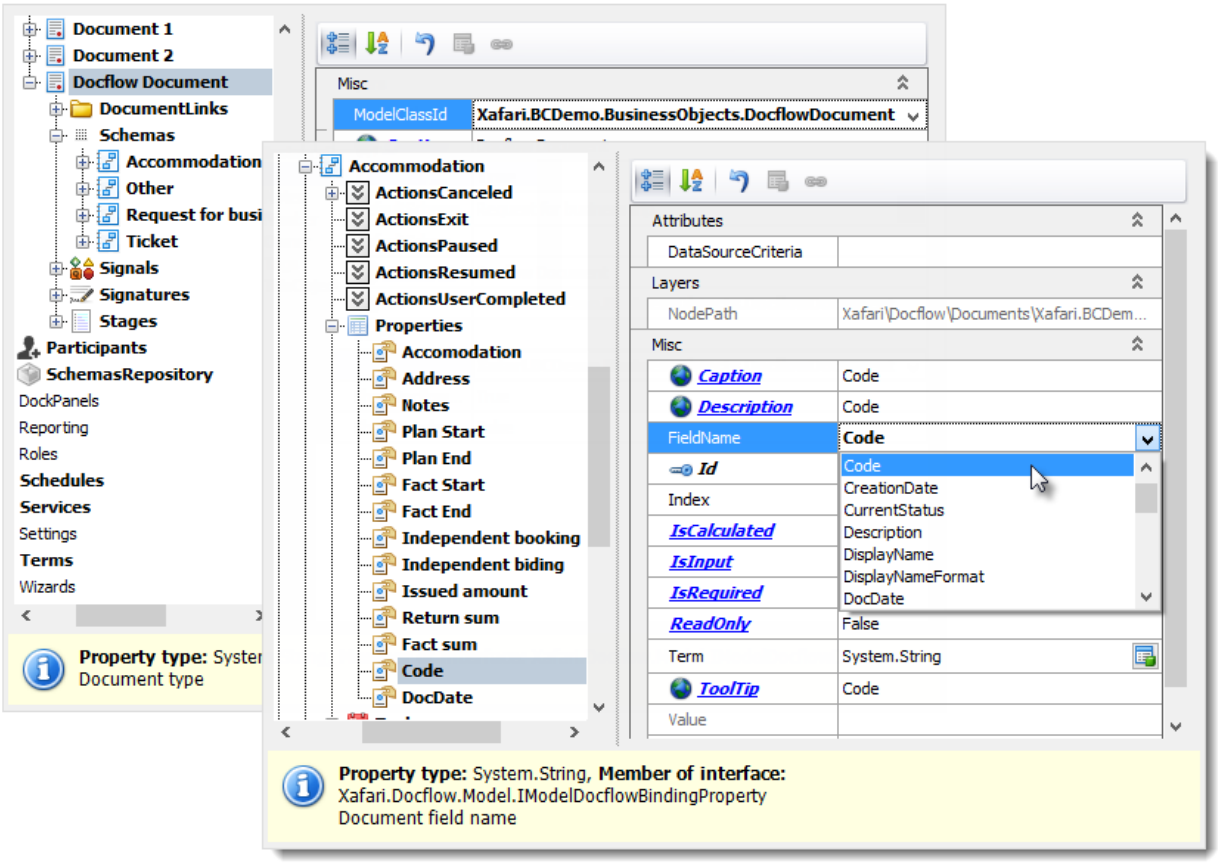
	moment when the system requires to set a value is the execution of the Task, which refers to this Property. The Task will not be completed while the value is empty.
ReadOnly	Boolean value indicating whether the user can edit the value or not.
Term	Refers to a predefined Term that specifies the data type and display options. See **Xafari|Terms** node.
Value	Specifies the default value or expression to calculate the default value.

Simple Property is an ancestor to other kinds of properties, each of which extends it with specific features.

Binding Property

This Property is a reference to the property of the persistent type that is associated with the Schema (more precisely, it is the type associated with the Document node containing the Schema). For example, for an **Accomodation** Schema, you can refer to one of the properties of the **Xafari.BCDemo.BusinessObjects.DocflowDocument** class, since this type is specified in the **ModelClassId** property of the **Docflow Document** node.

Binding Property allows the Schema to work equally with both persistent and dynamic properties.

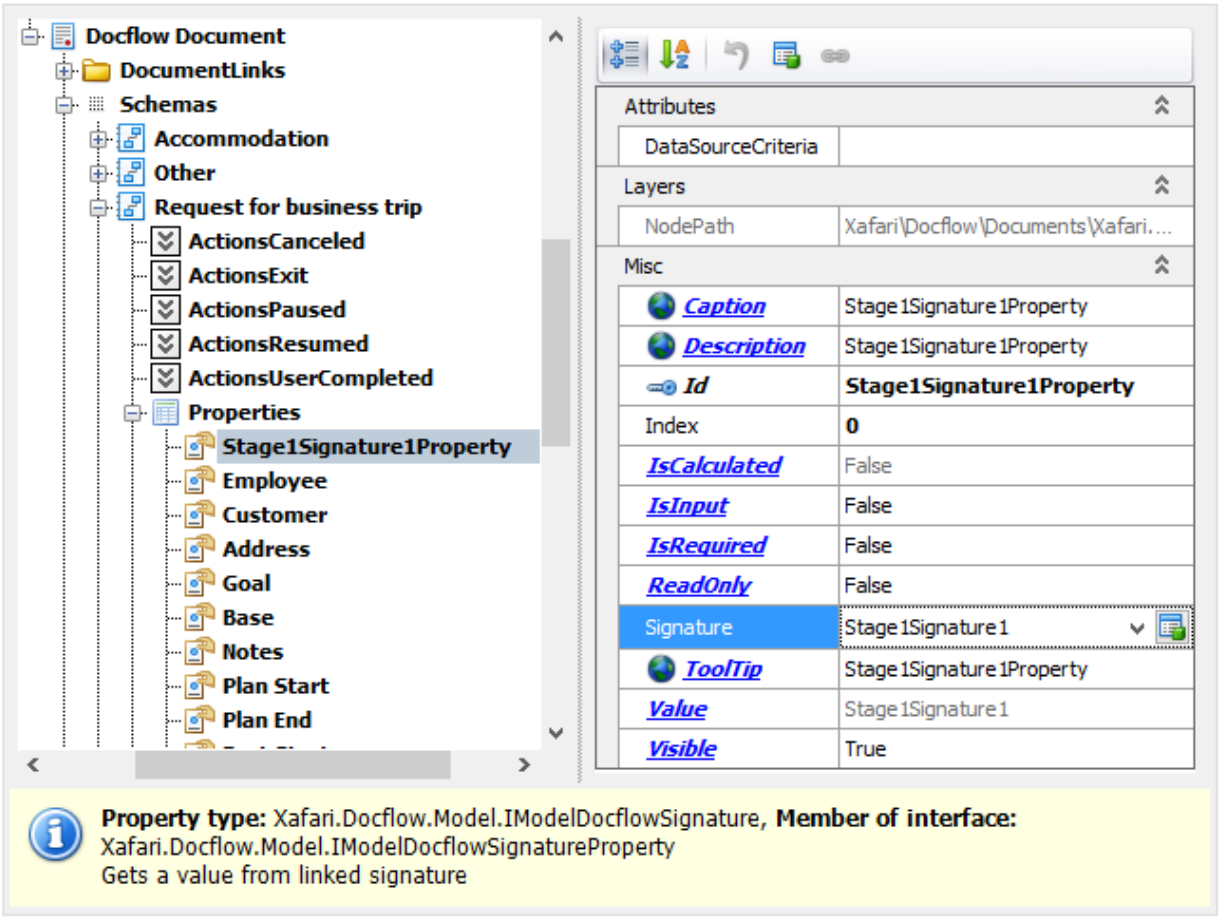


Additionally the Binding Property node exposes the **FieldName** property, it specifies the referenced field of the owner type. This node also has two predefined properties:

- **IsCalculated** set to **False** and blocked
- **IsRequired** set to False

Signature Property

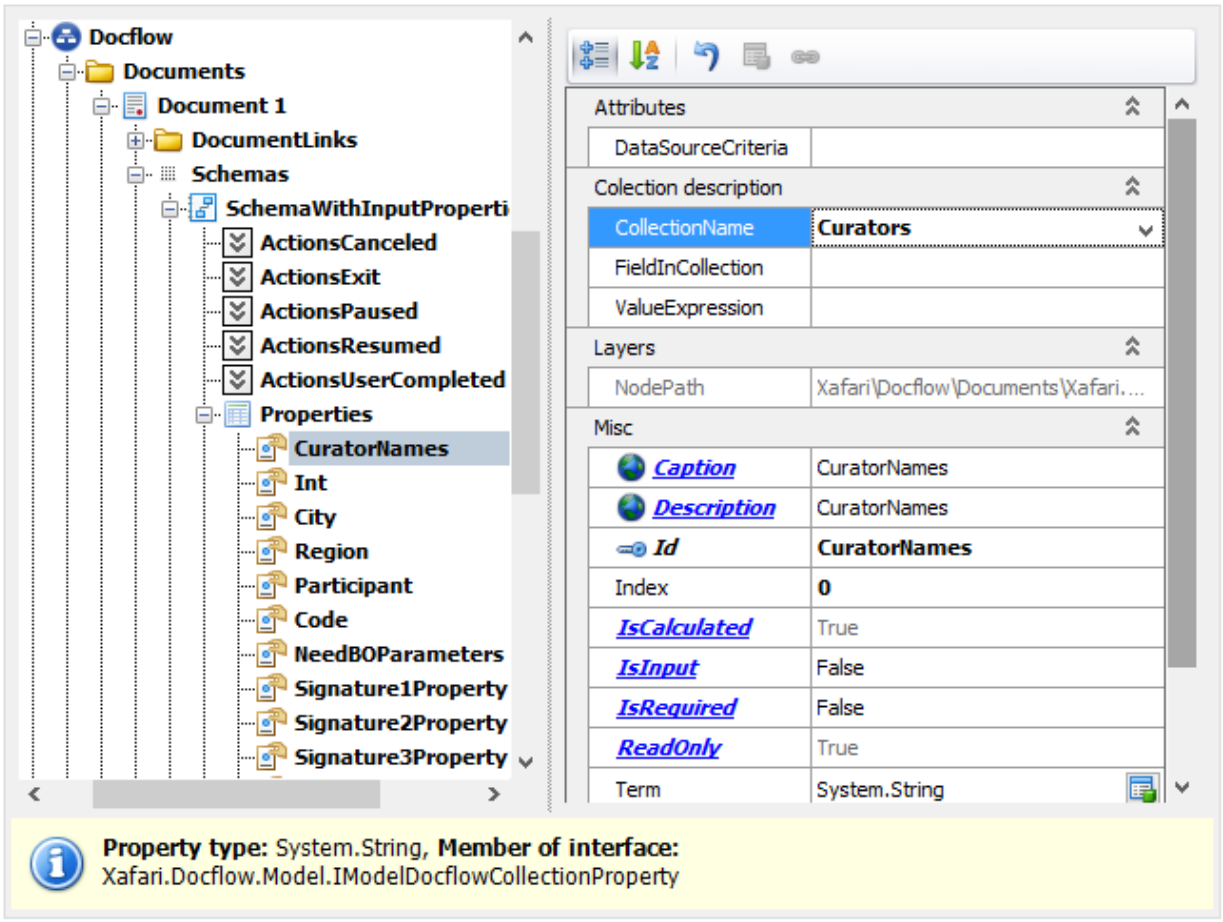
It is a reference to the Signature object.



The Signature Property node additionally exposes the **Signature** parameter, and, at the same time, this node does not have the **Term** parameter.

Collection Property

This Property returns a collection of objects or a value calculated based on the collection. This is not editable and is for internal use only, for example, for the circular generation of Tasks from a template and automatically assign the Executors. This is an advanced use case that is presented in the **SchemaWithInputProperties** of the **Document1** in the **BCDemo** app.



Properties:

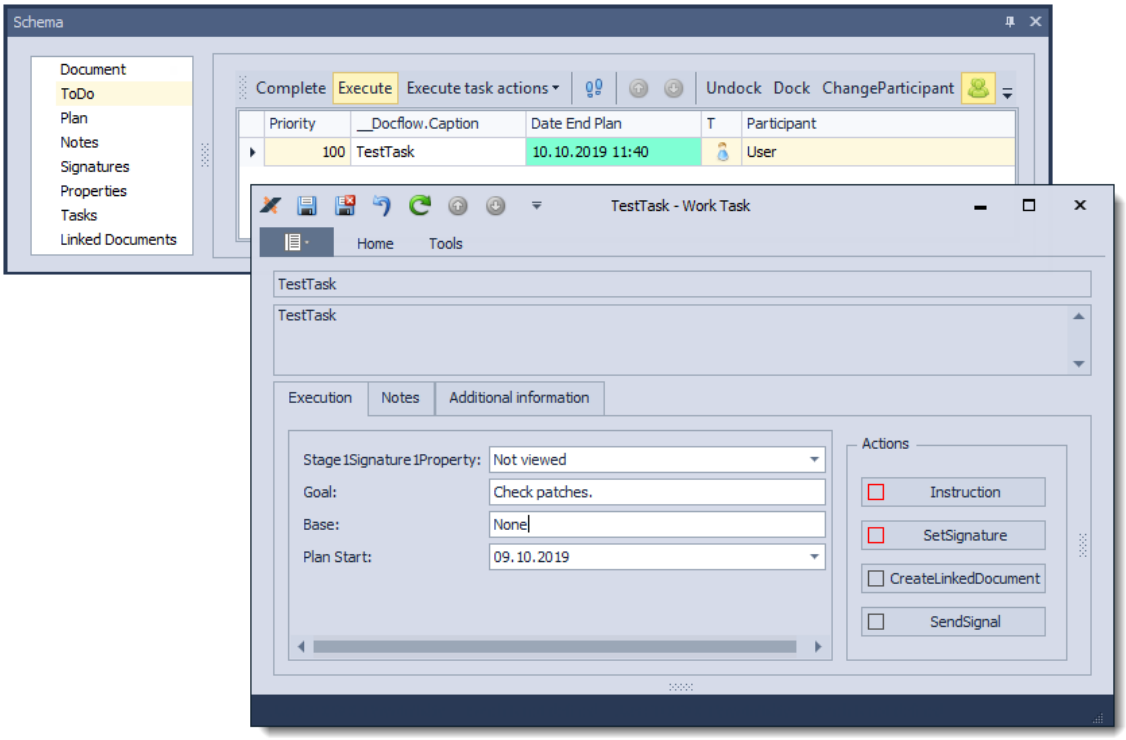
Property	Description
CollectionName	Specifies the collection of the Document type
FiledInCollection	Specifies the property of the objects, a new collection will be returned containing only the specified field
ValueExpression	Expression to be applied to objects in the collection to calculate a new value

The Collection Property node also has four predefined properties:

- **IsCalculated** set to **True** and blocked
- **IsRequired** set to **False**
- **ReadOnly** set to **True** and blocked
- **Visible** set to **False** and blocked.

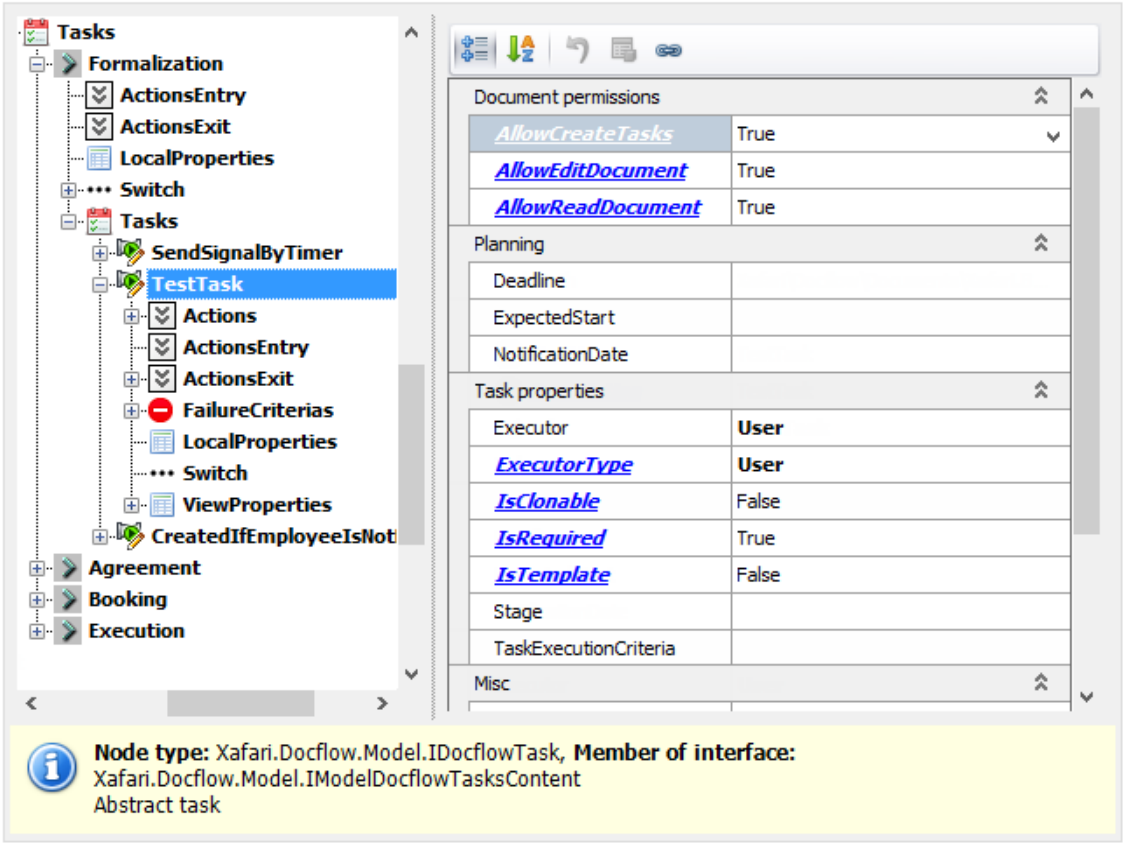
Docflow. Tasks

Task is a Docflow unit that transfers a Document object from a previous state to the next. The Task can be assigned to the user or to the system. The interface, functionality, and execution of the Task are described in the Docflow in Application topic.



Task

The Task includes a set of specialized components for flexible integration into the Document processing. Tasks can be combined into Task Groups, which simplifies management, allows you to define the same parameters once, for example, the **Deadline**. Image below shows **TestTask** node of the **Request for business trip** Schema in the **BCDemo** application.



Properties:

Property	Description
AllowCreateTasks	This property is fictitious now, it will be implemented in future. versions.
AllowEditDocument	This property is fictitious now, it will be implemented in future. versions.
AllowReadDocument	This property is fictitious now, it will be implemented in future. versions.
Deadline	Specifies the date by which the task should be completed.
ExpectedStart	Specifies the date when the task should be started.
NotificationDate	Specifies the date when the user will be reminded about the Task. There are three ways to set the **Deadline**, **ExpectedStart**, **NotificationDate**: • Set the date directly using the keyboard or popup calendar. • Specify the number of days from the start of the Task (or Group). • Use expression
Executor	Specifies who will be assigned the Task, this value depends on the **ExecutorType** property. In the case

	of a **User** or **Role**, the set of available values is limited to the **Participants** node (see Docflow Composition|Participants). The **User**, **Role** or **Property** objects can be specified or calculated at runtime.
ExecutorType	Specifies the type of Executor. The "**System**" value means that the Task will be executed automatically under the appropriate conditions. Other options are discussed in the Docflow Composition|Participants section.
IsClonable	Specifies whether the Task can be cloned.
IsRequired	A boolean value, the True value means that execution of the Task is required. It has effect only in parallel Task Groups. It is necessary to execute all required Tasks to complete a parallel Task Groups.
IsTemplate	Specifies whether the Task is a template
Stage	Binds the Task to the specified Stage.
TaskExecutionCriteria	The property contains an expression, which is evaluated at the time of the analysis of the model before generating the next pool of Tasks. The **False** value means that Task execution is not required and the Task will not be generated.
Index	The index determines the order of the Task within the parent Task Group, this is the order of execution for a group of sequential Tasks.

Collections:

Collection	Description
Actions	Actions provide elementary customizable operations, they are grouped into three categories. The main group is performed by the user in the predefined order. Additional Actions are performed by the system when appropriate events occur.
ActionsEntry	
ActionsExit	
FailureCriterias	The collection contains criteria items, these criteria are calculated when the Executor tries to complete the Task. The Task can be completed only if all criterias return **True**. The types of Failure Criterias and their settings are described below.
LocalProperties	A local Property is a reference to a global Property defined in a given Schema (or template). Local Properties allow the Task to store and use values other

	than those defined at the Document level. This is actively used when several instances of Tasks are dynamically created in runtime based on one template, each instance must have its own properties.
Switch	The collection allows to define custom scenarios for switching between Tasks. This allows you to change the main flow of Document processing depending on different conditions. The Switch is triggered when a Task is completed or a Signal arrives. If the **Switch** collection is empty or no condition is met, then the Docflow will simply proceed to the next Task. The types of Switches and their settings are described below.
ViewProperties	Specifies the Document Properties, that must be displayed in the View while the Task is executing.

FailureCriterias

There are two types of Criterias:

- Failure criteria
- Failure criteria wait signal.

The simple **Failure criteria** node exposes two required properties:

- **CriteriaExpression**: specifies the expression that determines the criterion.
- **ErrorText**: specifies the text message that will be displayed if the criterion is not satisfied.

The **Failure criteria wait signal** accumulates the facts of receipt of Signals from Linked Documents. When the Task is completed, the criterion is calculated taking into account the received Signals.

- **DocumentLink** and **Signal** properties specifies the Signal to be received.
- **CalculationType** property sets the way how to make a decision. This is significant for situations when the Signal comes from different instances of a Linked Document created dynamically. **AND** or **OR** determines whether to expect a Signal from all Documents, or at least one is enough. This allows to postpone the completion of the Task until some event occurs in all related Documents (or at least in one).
- **DocumentLinkCriteriaExpression** property may contain an arbitrary expression, as a rule, it operates with the state of the Linked Document. This expression takes precedence over **CalculationType**. If this property is set, then **CalculationType** is ignored, and the Failure criterion is set to **True** or **False** using this formula.

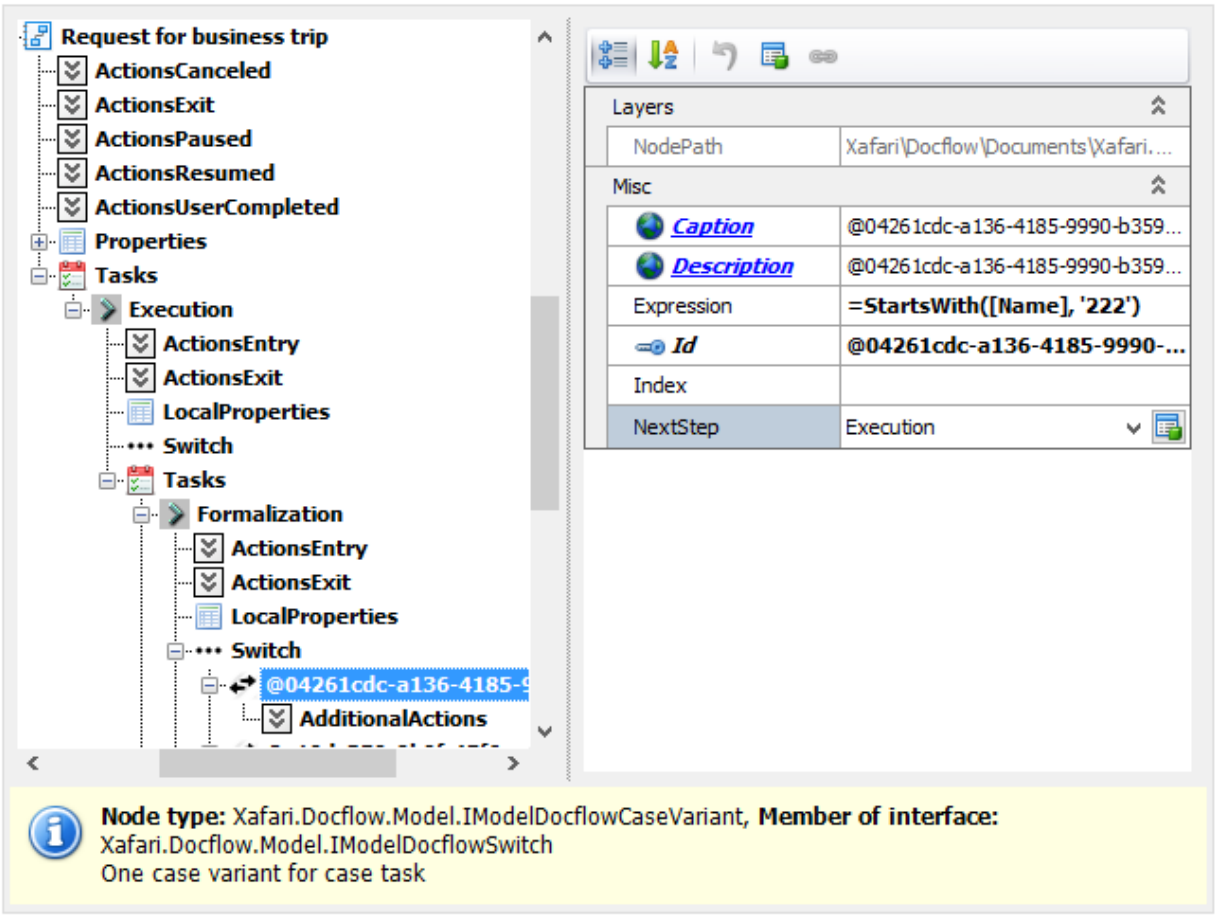
Switches

Depending on the triggering condition, there are three types of Switches:

- Case - GoTo
- Signal - Process
- Signal - GoTo

Case - GoTo

This type of Switch is triggered when the Task is completed, it executes additional Actions or proceeds to the specified new Task or both.



Properies and Collections:

Property or Collection	Description
Expression	The expression describes the criterion that triggers the Switch if **True** is received.
NextStep	Task or Group where to go. You can specify a Task from the current Group or another Group.
AdditionalActions	Contains additional Actions to be performed when triggered.

Signal - Process

This type of Switch is triggered when the specified Signal is received, it executes additional Actions only, it excludes the **NextStep** property and exposes the **Signal** property.

Signal - GoTo

This type of Switch is triggered when the specified Signal is received, it executes additional Actions and proceeds to the specified new Task (or Group).

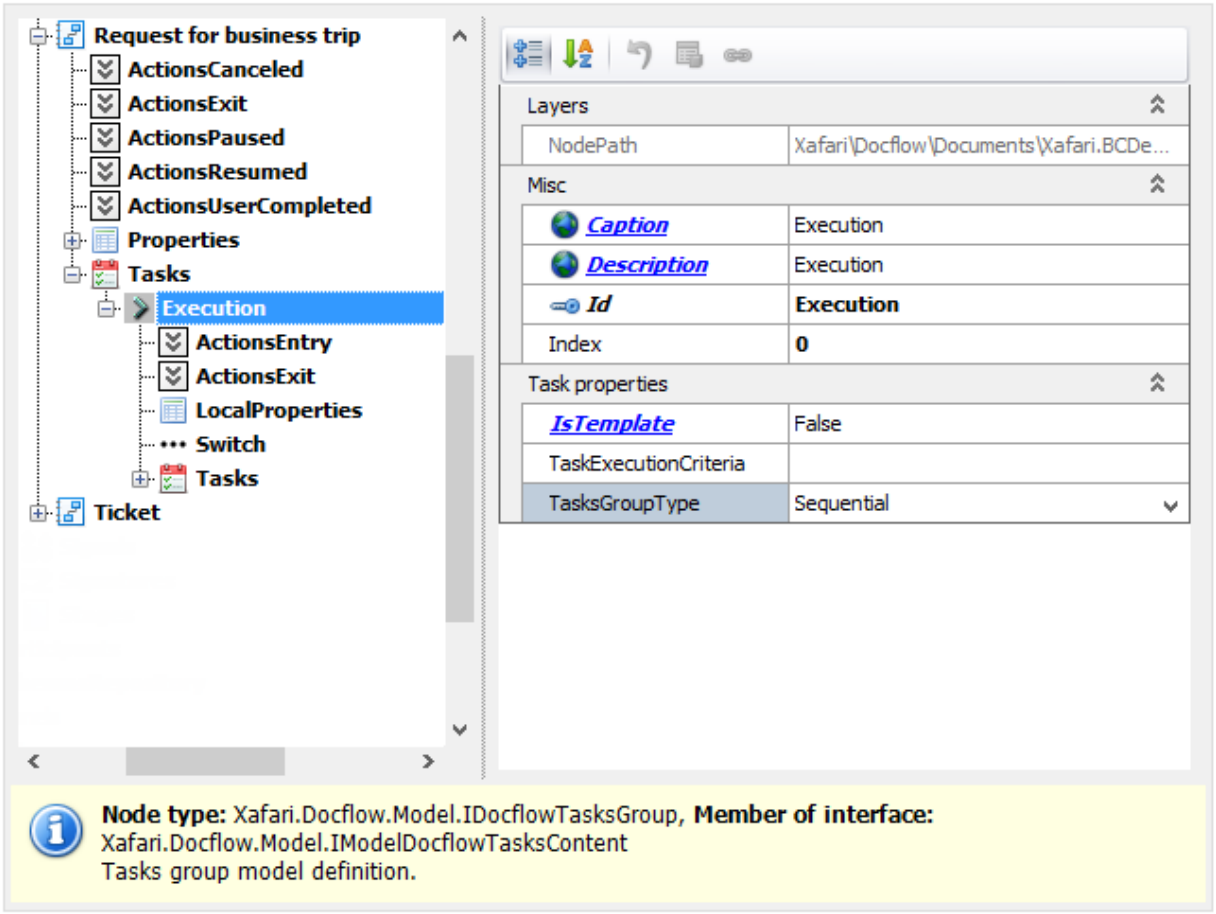
Properies and Collections:

Property or Collection	Description
Expression	The expression describes the criterion that triggers the Switch if **True** is received.
NextStep	Task or Group where to go. You can specify a Task from the current Group or another Group.
Signal	Indicates expected Signal.
TaskState	Sets the **State** property of the Task object before moving on to another Task. The possible states of the Task are declared in the **Xafari.BC.Tasks.TaskStates** enumeration. In most situations, the Docflow service manages this state, however, in the scenario under consideration, the status of the Task being abandoned remains uncontrolled. Before proceeding to the **NextStep**, set the status for the current Task. Default value is **2** (Succeeded).
AdditionalActions	Contains additional Actions to be performed when triggered.

Task Group

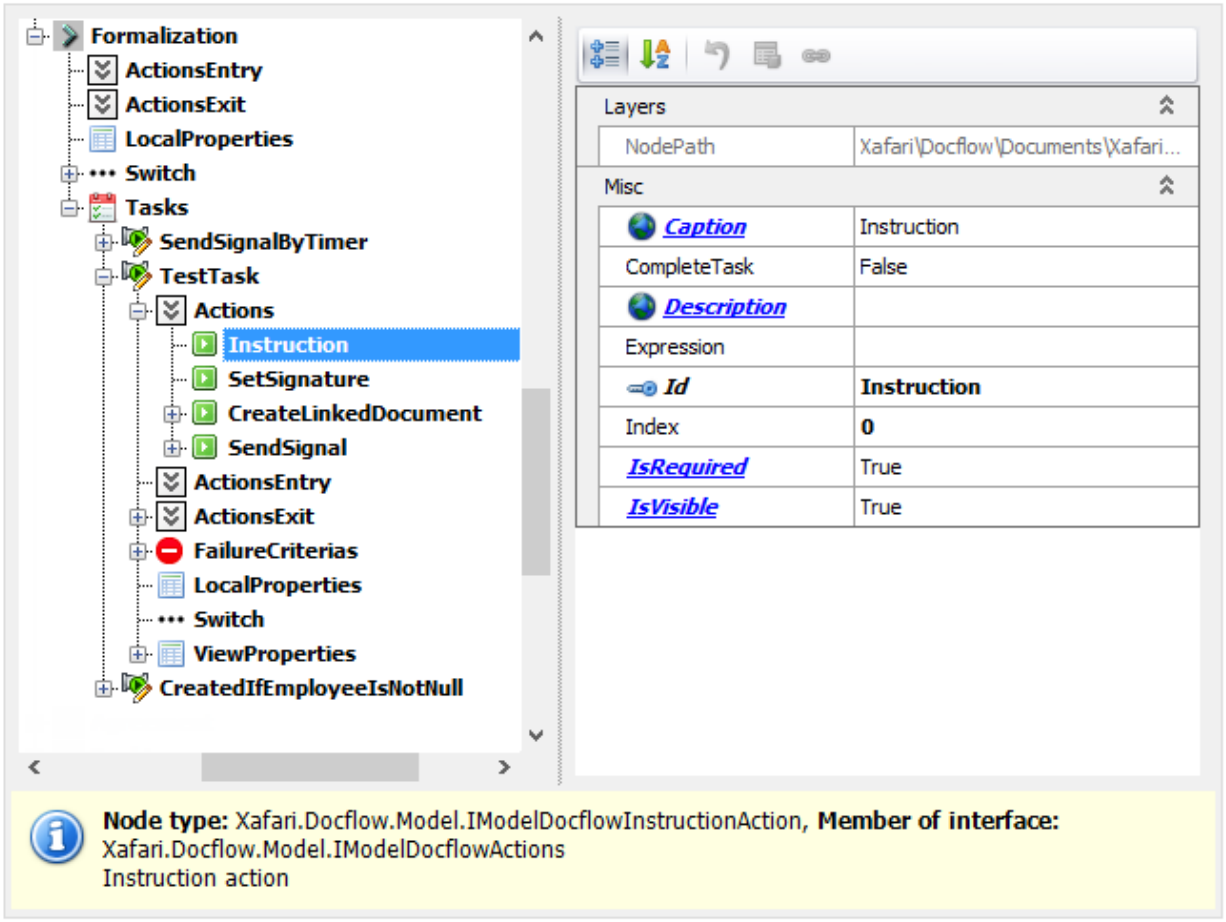
Task Group combines Tasks, Templates and other Groups into one structural unit, which greatly simplifies the development and perception of complex Schemas. It also allows to assign additional Actions at the entry and exit, describe a set of Local Properties, add common Switches. All these components are customized by analogy with a single Task.

The Group node exposes several properties already explained and has an additional **TaskGroupType** option, this determines the execution order of the nested Tasks.


Request for business trip
ActionsCanceled
ActionsExit
ActionsPaused
ActionsResumed
ActionsUserCompleted
Properties
Tasks
Execution
ActionsEntry
ActionsExit
LocalProperties
Switch
Tasks
Ticket
Layers
NodePath
Xafari\Docflow\Documents\Xafari.BCDe...
Misc
Caption
Execution
Description
Execution
Id
Execution
Index
0
Task properties
IsTemplate
False
TaskExecutionCriteria
TasksGroupType
Sequential
Node type: Xafari.Docflow.Model.IDocflowTasksGroup, Member of interface:
Xafari.Docflow.Model.IModelDocflowTasksContent
Tasks group model definition.

Docflow. Actions

Action in the context of Docflow provides an elementary operations from a predefined set, these operations can be performed by the user or the system. The system performs an Action when a certain event occurs, such Actions are called additional. An Action can be assigned to a Task, Group, or Template.



Any Action type has a basic set of properties listed in the table below:

Property	Description
CompleteTask	This flag indicates whether the Task should be completed after the Action has been executed.
Expression	This property allows you to describe an additional condition that determines whether the Action should be generated.
IsRequired	Indicates whether the Action is required, default value is **True**.
IsVisible	Indicates whether the Action will be displayed on the View.

The following describes the types of Xafari Docflow Actions and their properties.

Action - Instruction

It invokes a special View and displays the text from the current Task's **Description** property. This operation means that the Executor has read the instructions before proceeding the Task.

Action - Detail View

Invokes the specified Detail View, exposes the following specific properties:

- **AllowEditObject** specifies the View mode; the **True** value means that the Detail View will be displayed in the **Edit** mode; it is only actual for Web applications.
- CommitChangesOnViewClosing
- **ObjectKeyValue** specifies the object key, used to retrieve the object that will be displayed in the Detail View. It is possible to use an expression.
- **ObjectType** specifies the type of the object that will be displayed in the Detail View.
- **View** specifies the Detail View that will be displayed.

Action - List View

Invokes the specified List View, exposes the following specific properties:

- **BackFilter** specifies the filter criteria to be applied to the objects.
- **ObjectType** specifies the type of the displayed objects.
- **View** specifies the List View to be displayed.

Action - Send signal

Sends the specified Signal with attached parameters. It is possible to configure the frequency of sending.

Properties:

Property	Description
Signal	Signal to be sent, available Signals are listed in the **Signals** collection of the Document node.
CronExpression	This property contains the Cron expression that regulates the sending of the Signal on a Quartz schedule.

SendSignalParametersMapping collection provides a correspondence between the Shema Properties and Signal parameters. Signal parameters are intended to allow related Documents to exchange Properties values.

Action - Wait signal

Waits for a specific Signal from the linked Document.

This Action does not require the user to activate any visual control. When the Action's queue in the Schema arrives, execution begins automatically. The goal of its execution is to receive one or more Signals and calculate its state. The Schema is suspended until a positive result is obtained. In the simplest case, this means that the Shema will be suspended until the one expected Signal arrives. Then the Shema continue.

Properties

Property	Description
DocumentLink	Defines the linked Document from which the Signal is expected
Signal	Defines the type of expected Signal, available values are listed in the **Signals** collection of the Document node.
CalculationType	This property sets the way how to make a decision. This is significant for situations when the Signal comes from different instances of a linked Document. AND or OR determines whether to expect a Signal from all Documents, or at least one is enough.
DocumentLinkCriteriaExpression	Contains an expression that will be calculated at the moment the signal arrives, as a rule it includes Signal parameters that carry information about the state of the linked Document. This property takes precedence over **CalculationType**, if an expression is used, then **CalculationType** is ignored, and the system will make a decision based on the result of the expression.

ReceiveSignalParametersMapping Collection

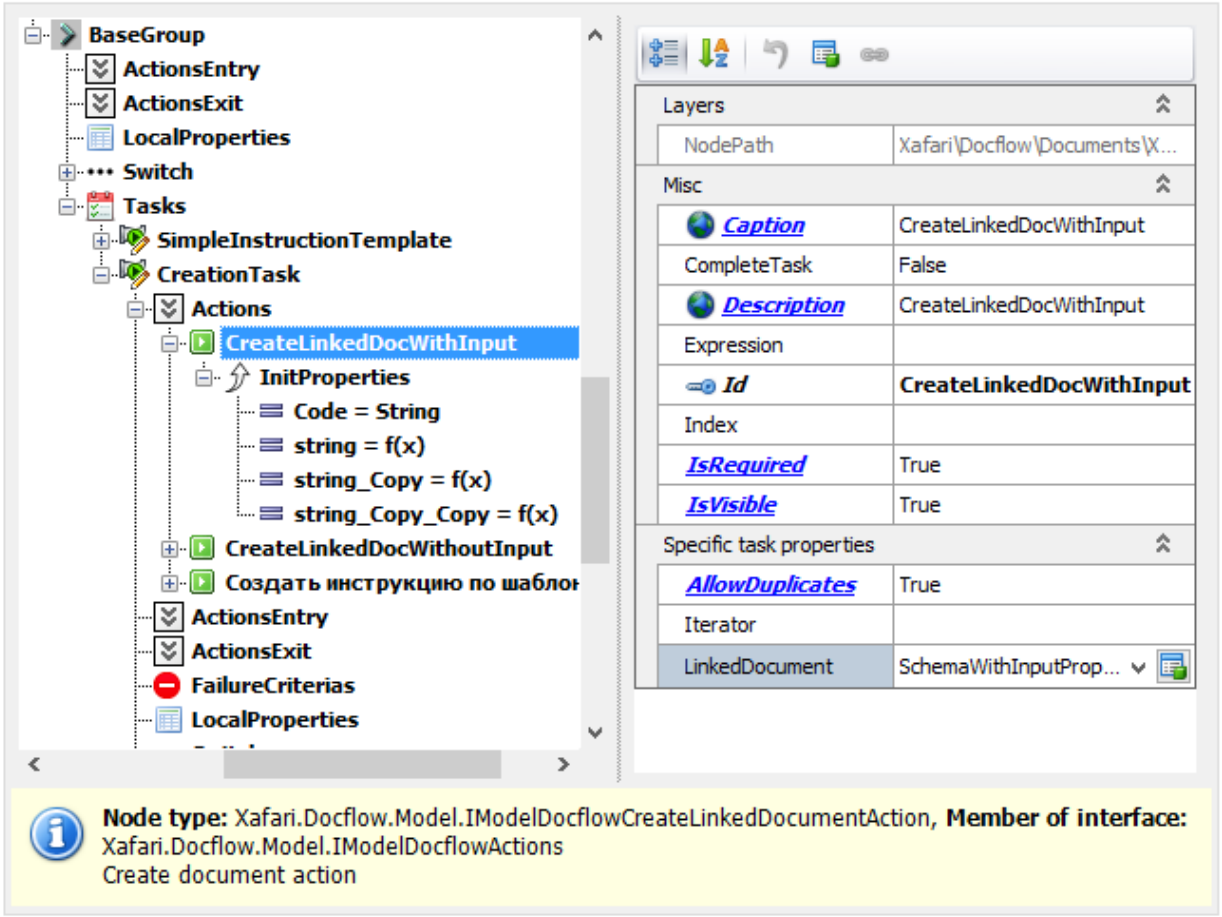
The collection maps the parameters of the expected Signal to the Document Properties, if the Task has Local Properties, the values will be entered into these Properties. Instead of directly matching types, the mapping mechanism uses Terms, this allows to bind a Signal parameter to a Property with a suitable type.

If the expected Signal comes from several linked Documents, then the parameters will be taken from the Signal, after receiving which the Action calculates a positive result, and the Scheme will continue.

Action - Create Linked Document

Creates a new linked Document.

An important feature is the ability to automatically initialize the required properties of a new Document. This is provided by the **InitProperties** collection. It is also possible to create several Documents of the same type at once, and the exact count will be calculated dynamically at the time the Action is executed.



Refer to the **Document2** document and examine the **CreationTask** of the **SchemaWithoutInputProperties** Schema. **BCDemo** app provides several Actions of this type.

The following table details the specific settings of the Create Linked Document Action.

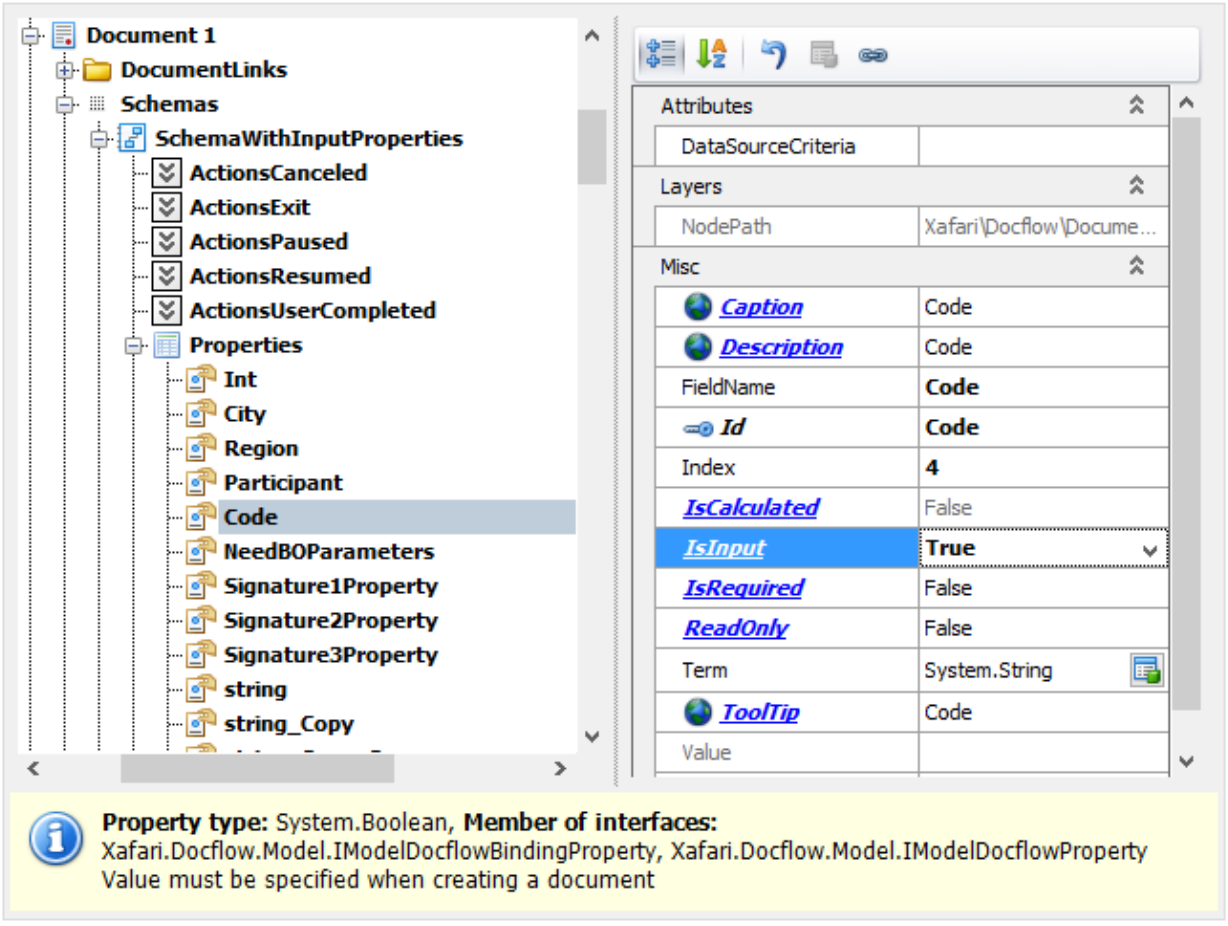
Properties

Property	Description
AllowDuplicates	Determines whether or not to create several identical Documents by re-executing.
Iterator	This property allows to create several new objects (in this particular context, this is Documents), and it also specifies the property by which they will differ. The **Iterator** parameter is actual together with the **InitProperties** collection described below.

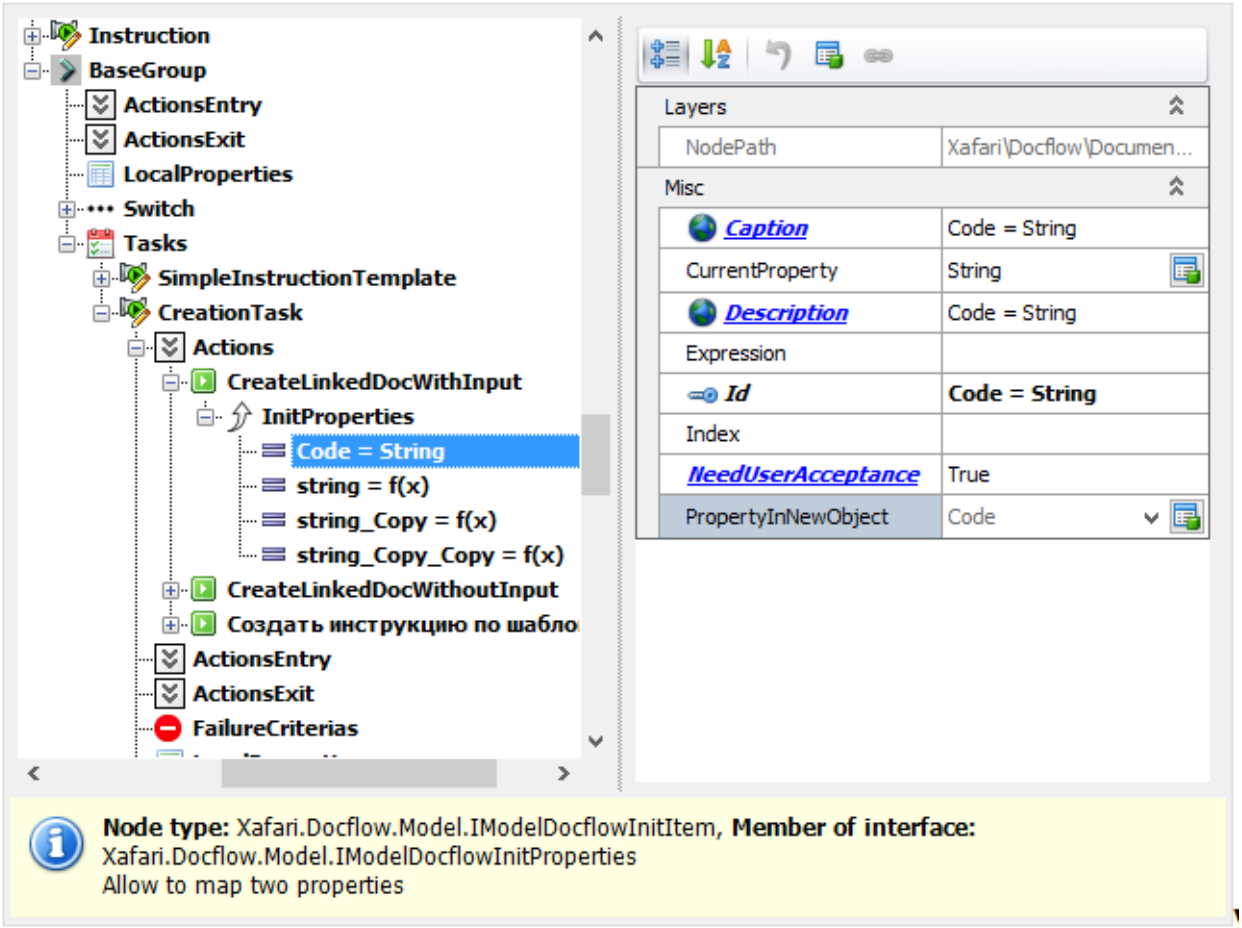
	The Schema of the primary Document must include the Collection Property (see Properties). In the **InitPropertis** collection, an item must exist that maps the Collection Property to the single Property of the new Document. It is understood that their types are compatible. This item should be specified in the **Iterator** parameter. Then Action will create as many new linked Documents as there are elements in the Collection Property of the primary Document at that moment. In accordance with the mapping, the property of each new object will be initialized by the element of the collection.
LinkedDocument	Specifies the Schema for the new Document to be created.

InitProperties Collection

InitPropertis collection is intended to initialize some Properties of the new linked Document. When designing a set of Schema Properties, the Property's **IsInput** parameter can be set to "**True**" value. This requires that the Property must be set when creating the Document object.



When configuring the Action's node, the Docflow service analyzes the specified Schema of the new linked Document and adds an item to the **InitPropertis** collection for each **IsInput** Property.



As you can see in the image above, the **PropertyInNewObject** parameter indicates the Property of the new Document that should be initialized, this value is read only.

The **CurrentProperty** parameter allows to specify the Schema Property of the current Document to initialize **PropertyInNewObject**. If their types are incompatible, then a "**null**" value will be applied.

If the **Expression** parameter is used, then the value of **PropertyInNewObject** will be calculated by the specified expression, and **CurrentProperty** will be ignored.

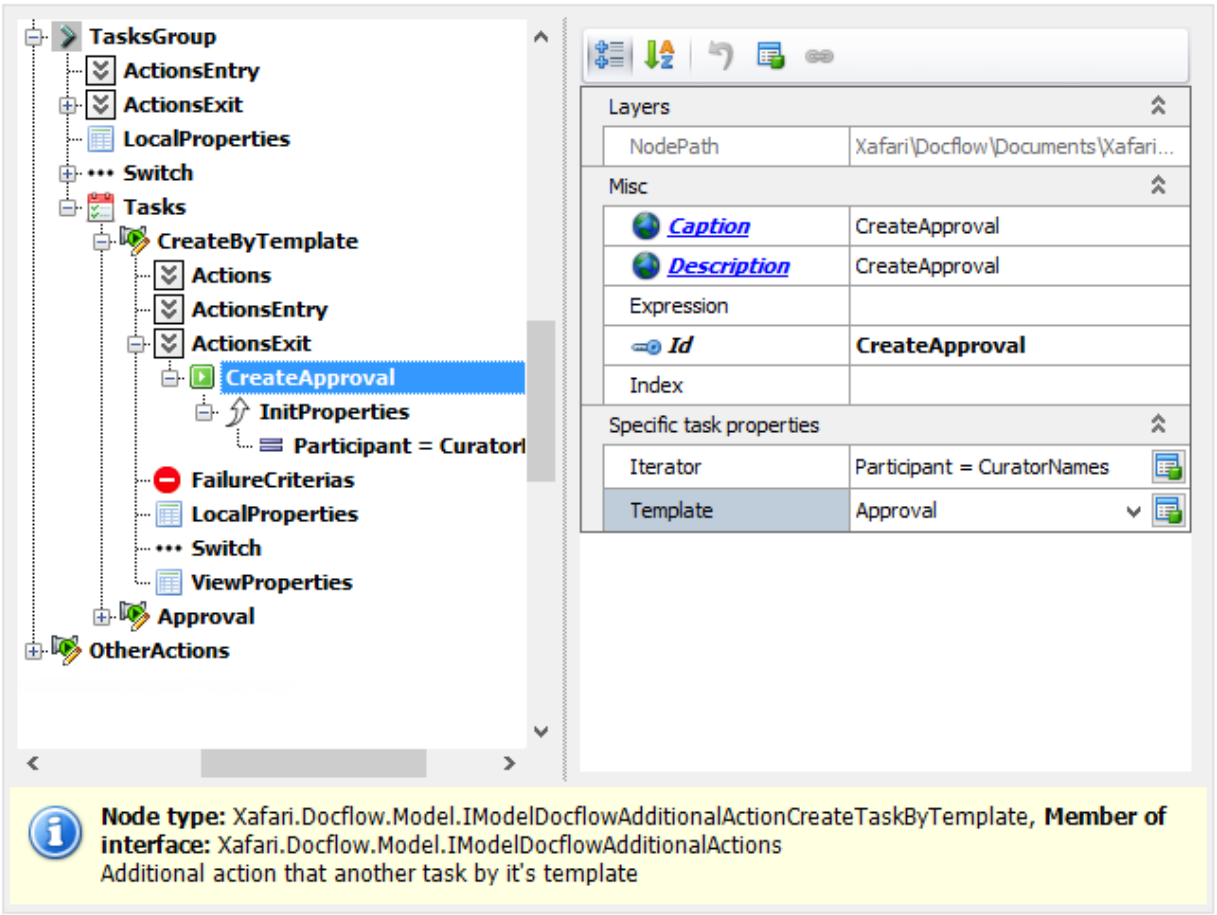
NeedUserAcceptance parameter indicates whether the user is required to confirm (or correct) the value at the time the object is created.

A special case is mapping a collection to **PropertyInNewObject**. The purpose of such an operation is to obtain as many new Documents as many items in the collection exist. To do this, the **CurrentProperty** parameter must specify the Collection Property from the Document Schema (see Collection Property in Properties topic). And the **Iterator** property should contain an **InitProperties** item in which this mapping is specified, see the **Iterator** description above.

Action - Create By Template

Creates an additional Task using the specified template.

Similarly to Create Linked Document Action, it is possible to initialize properties using the **InitPropertis** collection, as well as to obtain several objects using the **Iterator** parameter. The created Task takes place in the queue after the current one.



Refer to the **Document1** document and examine the **CreateByTemplate** Task of the **ShemaWithInputProperties** Schema.

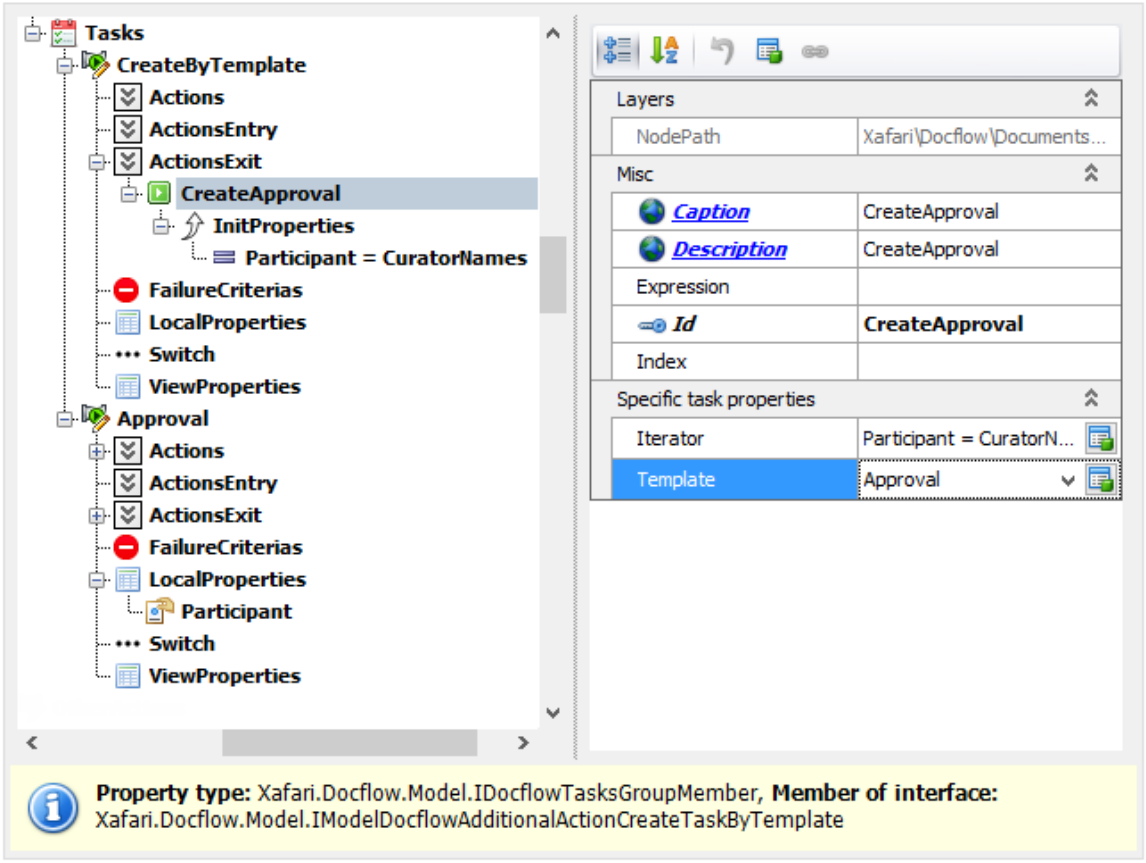
The table below lists the Action's specific settings.

Properties:

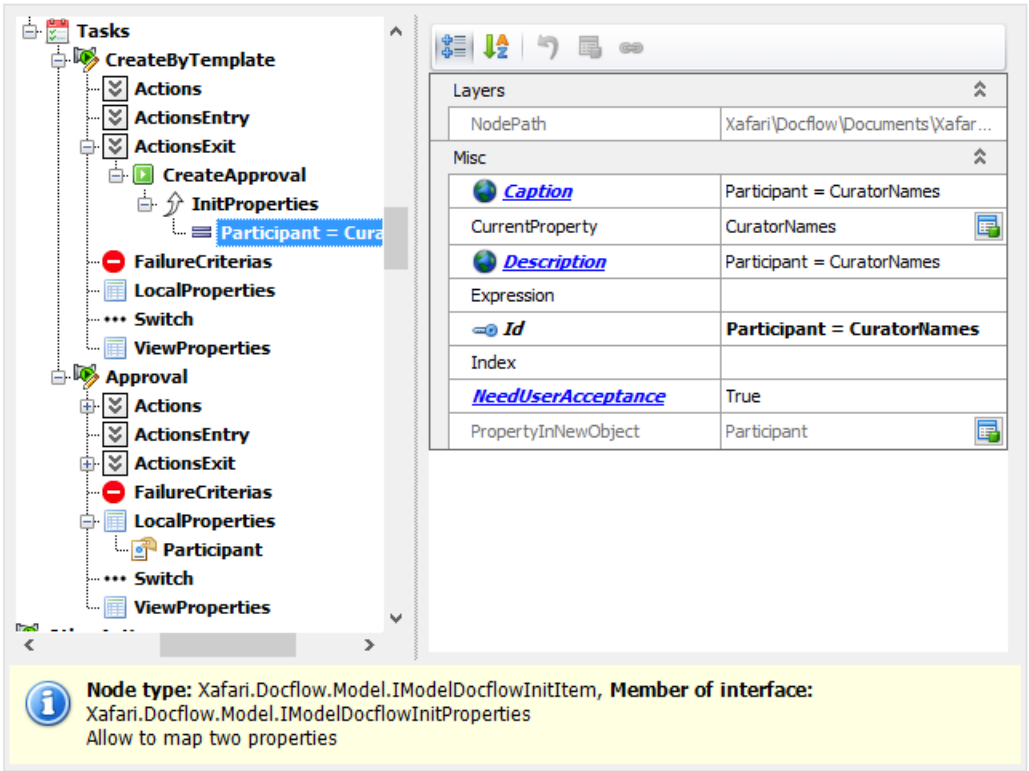
Property	Description
Iterator	This property allows to create several new objects (in this particular context, this is Tasks), and it also specifies the property by which they will differ. The **Iterator** parameter is actual together with the **InitProperties** collection described below. The concept of using this parameter is completely similar to Create Linked Document Action.
Template	This parameter refers to an existing Task, which will be used as a template for a new one. Tasks within the current group are available for this.

InitProperties Collection

InitPropertis collection is intended to initialize Local Properties of the new Task. Local Properties of the Task are listed in the corresponding node, see Tasks topic.



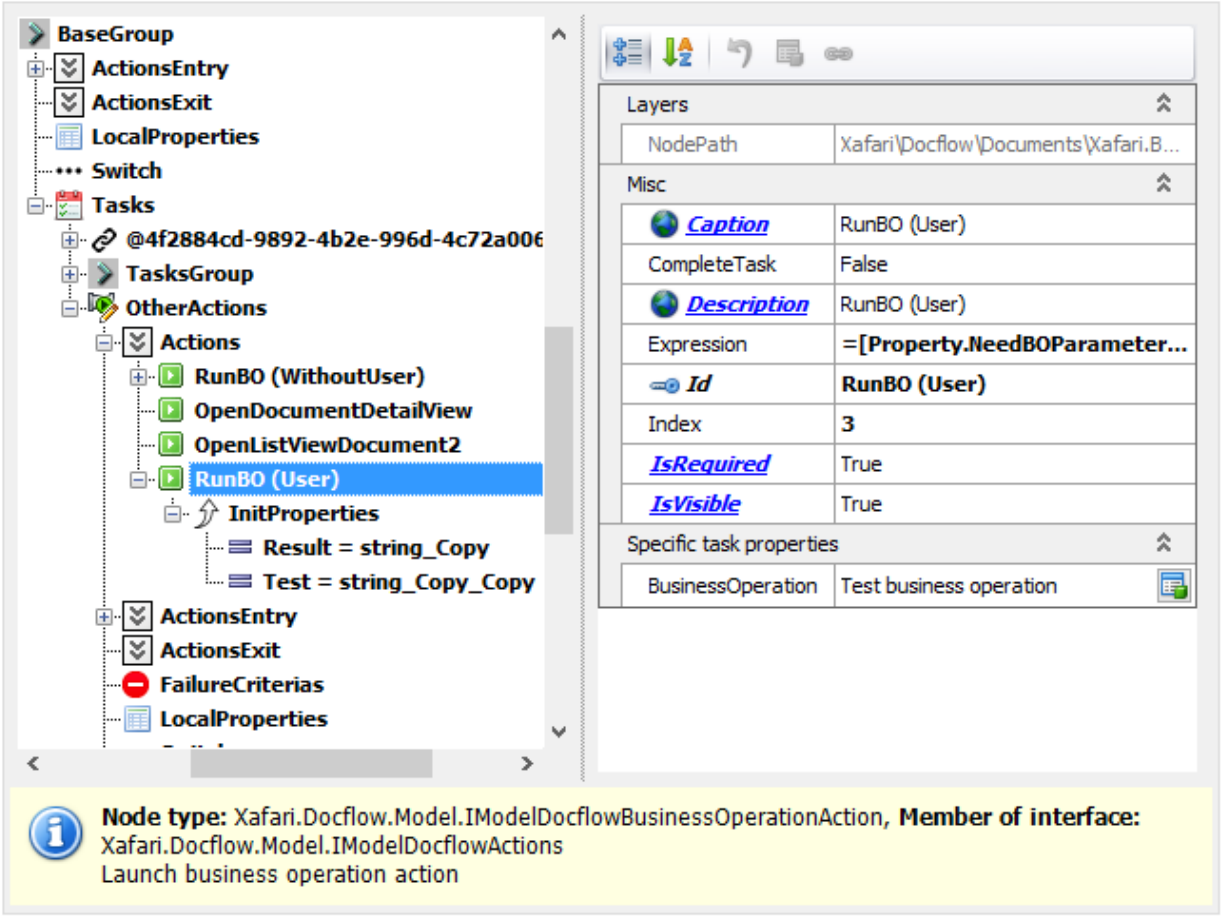
When configuring the Action's node, the Docflow service analyzes the specified Template of the new Task and adds an item to the **InitPropertis** collection for each Local Property.



Configuring items in the **InitProperties** collection is completely identical to Create Linked Document Action described above.

Action - Business Operation

Starts the specified Business Operation; if this operation has input parameters, they will be initialized via the **InitProperties** collection.



Refer to the **Document1** document and examine the **OtherActions** Task of the **ShemaWithInputProperties** Schema.

The **BusinessOperation** parameter specifies the Business Operation to be started.

The **InitProperties** collection contains associations between the Schema Properties and parameters of the Business Operation. An **InitProperties** item is automatically generated for each parameter of the Business Operation, the item sets the start value for the respective input parameter, and after completion of the Business Operation the output parameters are sent to the Schema.

Action - ExecuteAction

Executes the specified XAF Action, exposes two specific parameters:

- Action
- Controller

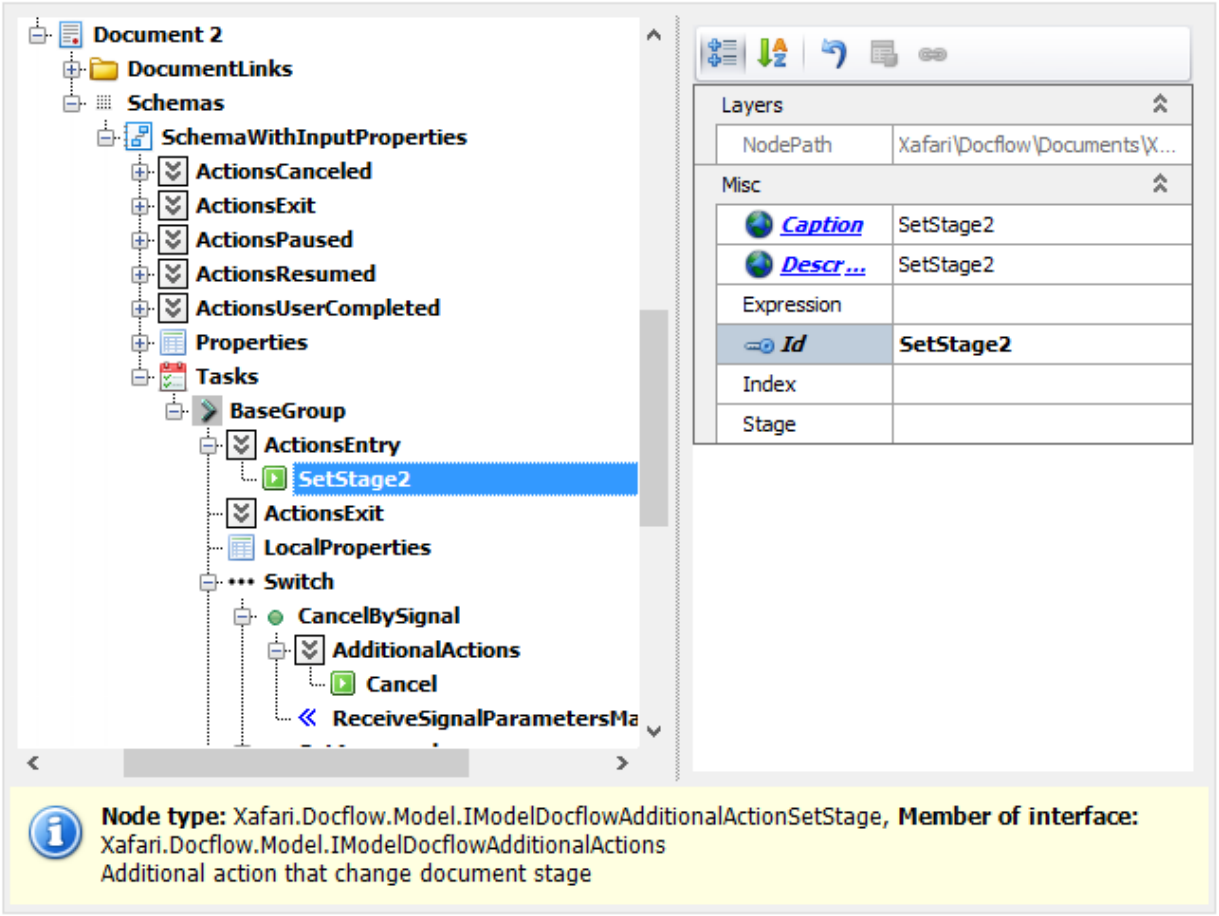
Action - Approve signature

Invokes a special window to approve or reject the specified Signature, write a comment. The final window configuration depends on the availability of additional statuses for the Signature. It exposes the following specific properties:

- Signature: specifies an item from the **Signatures** collection of the Document structure.
- NoActionCaption: sets a text for the **No** button.
- YesActionCaption: sets a text for the **Yes** button.

Docflow. Additional Actions

In addition to the main Actions represented in the Task interface, there are additional Actions that are executed only by the system when certain events occur. Additional Actions can be assigned to Task, Task Group, Schema, link to the Template In Repository, etc.



There are a number of events in the Scheme when the system can execute additional Actions:

- Schema Canceled
- Schema Exit
- Schema Paused
- Schema Resumed
- Shema Completed
- Task Group Entry
- Task Group Exit
- Task Group Switch
- Task Entry
- Task Exit
- Task Switch

The set of additional Actions intersects with the set of main Actions, and the basic options are the same. The following table lists the available positions.

Action	Description
Set stage	Sets the value of the specified Stage
Execute system action	Executes the system Action that controls the Schema: Cancel, Pause, Complete.
Send mail	Sends mail to the specified recipient
Send signal	Sends the specified Signal
Set status	Sets the value of the specified Status of the Document
Reset signature	Resets the state of the specified Signature
Business operation	Starts the specified Business Operation
Set property	Sets the value of the specified Property
Set signature	Sets the value of the specified Signature
Create by template	Creates an additional Task using the specified template.