



Nirva WORKFLOW Service

Document Version: 1.36

Table of Contents

Overview	7
Logical components	7
Workflow	8
Process	9
Activity	10
Link	13
Variables	14
Triggers	14
Data	15
Correlation tables	15
Audit	15
Engine	15
Processing	16
Creating a process	16
Starting a process	16
Ending a process	17
Process purge	17
Process recovery	17
Running activities	17
Managing time out and wait activities	18
Scripts	18
Transition condition	18
Join condition	19
Installation	20
Configuration	22
Workflow	22
Workflow list	22
Creating a new workflow	23
Starting and stopping a workflow	24
Graphically configuring a workflow	24
Removing a workflow	24
Exporting a workflow	24
Importing a workflow	24
Modifying workflow parameters	25
General workflow parameters:	25
Activities	25
Links	29

Correlation tables	32
Production.....	33
Search	33
By date	33
By process ID	35
By correlation table.....	35
Detail process data.....	35
Activities	36
Links	37
Correlation data.....	37
Associated data.....	37
Variables	38
Auditing	38
Creating a new process.....	38
Graphical configuration	39
Interface organization	40
Toolbar.....	40
Title area.....	40
Drawing area	40
Location area.....	41
Configuration area.....	41
Message area.....	41
Toolbar.....	41
Configuration.....	42
Workflow parameters.....	42
Information group	43
Drawing area group.....	43
Correlations	43
Activity parameters.....	44
Information group	46
Procedure group.....	46
Triggers group	46
Evaluation group	46
Join group.....	46
Time outs group	47
Link parameters.....	47
Information group	47
Transition group	47
Drawing area.....	48
Activities.....	48
Autostart indicator	48
Activity type indicators.....	49
Action icons	49
Resize anchor	49
Moving activities	49
Adding activity	50
Deleting activities	50

Links	50
Adding links	50
Moving links.....	50
Deleting links	50
Area Zoom.....	50
Architecture	52
Workflow servers	52
SAN.....	53
Work servers.....	53
Interface servers	54
External applications.....	55
Example	56
Description	56
Installation.....	57
Prerequisites.....	57
Installation.....	57
Starting the application	58
Using the application	58
Analysis.....	63
Input.....	63
Index	65
Code	65
Workflow.....	66
Validation	68
Code	68
Workflow.....	70
Ok	71
Code	71
Workflow.....	73
Reject.....	73
Code	73
Workflow.....	75
Other modules	75
Init and cleanup code	75
Triggers	76
Listeners.....	76
Reference.....	77
Classes	77
Error codes	77
WORKFLOW Class	77
PROCESS Class	78
ACTIVITY Class	78
LINK Class.....	79
CORRELATION Class.....	79
Permissions	80
Commands.....	80
WORKFLOW class.....	80

CREATE.....	81
DISABLE.....	82
ENABLE.....	82
EXPORT.....	83
GET_PARAM.....	84
IMPORT.....	85
LIST.....	85
REMOVE.....	87
SET_PARAM.....	87
VERIFY.....	88
PROCESS class.....	89
AUDIT.....	89
CLOSE_REGISTRY.....	90
CREATE.....	91
END.....	92
INFO.....	92
OPEN_REGISTRY.....	94
PURGE.....	95
RECOVERY.....	96
RESET.....	97
RUN.....	98
ACTIVITY class.....	98
COMPLETE.....	98
CREATE.....	99
ERROR.....	101
GET.....	102
GET_PARAM.....	103
LIST.....	104
REMOVE.....	105
RESTART.....	106
SET_PARAM.....	107
SET_TIME_OUT.....	108
TIME_OUT.....	109
LINK class.....	110
CREATE.....	110
GET_PARAM.....	111
LIST.....	112
REMOVE.....	113
SET_PARAM.....	114
VARIABLE class.....	115
GET.....	115
LIST.....	116
SET.....	117
REMOVE.....	117
PRODUCTION class.....	118
SEARCH.....	118
CORRELATION class.....	120

CREATE_ENTRY	120
CREATE_TABLE	121
GET_PIDS	122
GET_VALUES.....	122
LIST_TABLES.....	123
REMOVE_ENTRY	124
REMOVE_TABLE	125
SET_TABLE_PARAM.....	125

Overview

The workflow service is a Nirva service that implements the logic of workflow scenarios for managing long running transactions.

The Nirva workflow service helps to architect an application from a business point of view. The workflow supports the business logic that is coded into activities. This business viewpoint only displays the different activities, their status and how they are organized together to meet requirements.

The service provides all the environment and functionality to control workflow processes and their various states. The execution of the activities is not part of the workflow and remains controlled by Nirva applications by using standard Nirva features. As usual, Nirva provides all the necessary functionality to start and run activities directly or in the background, scheduled or not, to wait for events, to send and receive messages, etc.

This allows the sharing of a complex business process on different servers, optionally having dedicated servers for some activities. The processing of some workflow activities can also be controlled by external software that can communicate with the Nirva workflow using client connectors.

The main features are:

- Workflow definition and deployment.
- Logic defined around activities and links.
- Unlimited number of workflow processes (only limited by disk size).
- No need of external database.
- Auditing.
- Correlation data for retrieving workflow processes using business data.
- Possibility to associate business specific data with process or activities.
- Web configuration, administration and history.
- Deployment facilities.

Logical components

The workflow defines the following logical components:

Workflow	Contains all the components allowing the definition and execution of a business process with a given logic.
Process	A process is a workflow instance. It is a workflow case created by an external event. For example: for an order management workflow, a new process is created each time a customer submits an order. Each process is identified by a unique key, a process ID (PID) generated by the system.
Activity	Unit task of a workflow. For example “document composition” or “invoice validation”. An activity receives a status.
Link	Logical link of 2 activities. A link has a source activity and a target activity. An activity has outgoing links and incoming links. A link receives a status of active or inactive.
Variables	A variable is a string having a unique name and a value. Variables can be defined at process or activity levels. The variables can be evaluated in the scripts that define the business logic in order to decide what subsequent activities should run.
Data	Any process has a dedicated persistent container that can keep any kind of data associated to it.
Correlation tables	The correlation tables allow the launch of retrieving processes using business data. For example: for a banking workflow, a correlation table can be used to retrieve workflow processes from the customer or account numbers.
Audit	All data related to workflow status change are kept on disk and can be displayed from dedicated screens.

Workflow

A workflow is made of the following components:

- Name Uniquely identifies the workflow.
- Description Free text that describes the workflow.
- Activity table List of activities defined for the workflow.
- Link table List of links between the workflow activities.
- Correlation tables Correlation tables allowing retrieving a workflow process from business data.
- Base directory The directory where the system writes and keeps all workflow data.
- Retention delay Number of seconds to keep the terminated process in the system. The default is 0 (process immediately removed when terminated).
- Start activity trigger A Nirva procedure called each time an activity starts. This can be useful for doing some external reporting.
- Stop activity trigger A Nirva procedure called each time an activity stops.

- **Purge trigger** Name of a Nirva procedure called when the process is removed from the system.

All data related to a particular workflow is stored on the file system in the workflow base directory. This allows keeping and recovering any process in case of crash and sharing the data with a cluster server to avoid synchronization.

A workflow can be packaged and installed on a target machine. The service also provides some import and export facilities in XML format of the workflow definition.

A workflow can be enabled or disabled. Any workflow definition modification can only be made if the workflow is in disable state.

Process

A process is an instance of a workflow (workflow case). It is created by a dedicated command from an external event (ex.: when a file arrives in a directory) or from a manual user request (ex.: a customer starts an order).

A process is uniquely identified by a string (PID) generated when the process is created.

A process terminates when an activity of type End or Error is started (see Activity).

All data related to a process is kept on disk until the process terminates and is purged.

The purge mechanism allows eliminating all terminated processes after a given retention delay.

A process receives a status that can take the following values:

- **INACTIVE** The process has been created but is not started.
- **RUNNING** The process is running.
- **COMPLETED** The process terminated successfully.
- **ERROR** The process terminated with error.
- **STOPPED** The process has been explicitly stopped.
- **DELETED** The process is ready to be deleted but has not been yet deleted (retention delay expired).

A process also receives a global status identifying if the process is active or terminated. A terminated process is a candidate to the purge mechanism that removes it after a defined retention delay. A terminated process can be terminated with success (global status is 1) with error (global status is 2) or can be stopped (global status is 3).

Compatibility issue: for process created with a workflow service version prior to 3.06, the global status takes only values 0 (active) and 1 (terminated).

Some variables can be defined at process level. These variables can be evaluated when testing the conditions for starting an activity.

The workflow can manage a very large number of processes. This is limited only by available disk space. The data storage is organized with performance optimization in mind and to avoid some operating system

considerations around the number of files and directories. Accessing a process is immediate when its PID is known. When not known, the correlation tables can be used to retrieve a PID from business information.

The workflow can run different processes in parallel but all access to a given process are synchronized automatically. This avoids any conflict when working on a process.

Activity

An activity is a single task of business logic executed in the context of a process. The activities can be linked together with links.

The code of an activity is external to the workflow, although the workflow controls its execution. An activity can be executed by the workflow itself (synchronous activities), by a task of the Nirva scheduler, by a Nirva listener, by a Nirva client, by a web interface or by any external application using a Nirva connector. Several activities can be executed in parallel.

An activity can receive the following status:

- **INACTIVE** The activity is inactive. It has not been activated (never ran) or has been inactivated.
- **READY** The activity is a Queue activity queue (asynchronous) and has been placed in a queue.
- **RUNNING** The activity is being executed.
- **COMPLETED** The activity has been successfully completed.
- **ERROR** The activity has been completed with error (error reason attached).
- **TIMEOUT** The activity has not been executed during the requested time.

There are different types of activities:

- **Invoke** Synchronous activity. A Nirva procedure is attached to the activity and is automatically executed when the activity starts. When running, the status subsequently turns to RUNNING state and then to COMPLETED or ERROR according to the result of the procedure.
- **Queue** Asynchronous activity. When the workflow starts a Queue activity, it is placed in a queue that can then be monitored by a listener or any application or user interface. When the activity starts, it goes to READY state and then to the RUNNING state when it has been retrieved from the queue. It remains in this state until the listener that executes it signals the completion (success or error) to the workflow or until the optional time out occurs. The final state will then be COMPLETED, ERROR or TIMEOUT. The Queue activity may have an optional time out and an execution time out. The time out concerns READY state period and the execution time out concerns the RUNNING state period.
- **Receive** Asynchronous activity. A Receive activity only waits for an event to occur. During this time, it is in the RUNNING state. The event is signaled to the workflow with dedicated

command (error or completed signal). As for a Queue activity, a time out can be set. The final state will be COMPLETED, ERROR or TIMEOUT.

- **Wait** Asynchronous activity. A Wait activity just waits for a given time or until a given date and time. It remains in the RUNNING state until the deadline is reached. It then goes to the COMPLETED state.
- **Block** Synchronous activity. A Block activity defines a group containing other activities. Activities that are part of a Block receive the Block name as parent name. While inside the Block, any activity in the Block is in RUNNING state. The Block will be terminated when a End or an Error activity is reached inside the block. At this time the Block activity goes to the COMPLETED or ERROR state.
- **End** Synchronous activity. The End activity allows leaving the block activity or the process with success. If the activity has a parent, the parent is a block activity that is then terminated with COMPLETED status. If it does not have any parent, the process itself is stopped when the End activity is executed.
- **Error** Synchronous activity. The Error activity allows leaving the block activity or the process with error. If the activity has a parent, the parent is a block activity that is then terminated with ERROR status. If it does not have any parent, the process itself is stopped when the Error activity is executed.



Activity of type Invoke should be reserved only when a real synchronous activity is needed or when the processing is very fast. For example for deciding which screen to display to a user. Otherwise queue activity is a better choice because it offers more flexibility and scalability.

The same activity can be executed several times in the same process or block. So the system maintains an execution counter for each activity. This counter is reset when the activity is part of a block and the block starts.

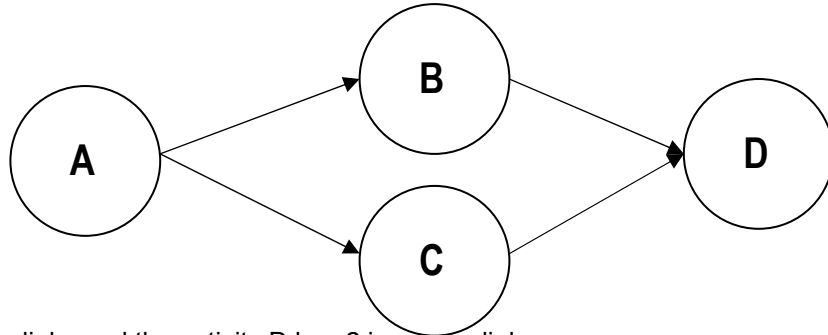
An activity is defined using the following parameters:

- **Name** Uniquely identifies the activity. Cannot contain any space or special character.
- **Description** Free text that describes the activity.
- **Parent** Name of the parent block activity if the activity is part of a block.
- **Type** Activity type. This can be "INVOKE", "QUEUE", "RECEIVE", "WAIT", "BLOCK", "END" or "ERROR".
- **Procedure** Procedure name to be executed for an Invoke activity. This parameter has no meaning for other types of activities.
- **Start trigger** A Nirva procedure called each time the activity starts. The trigger procedure defined at activity level is always executed after the trigger procedure defined at workflow level.

- **Stop trigger** A Nirva procedure called each time the activity stops. The trigger procedure defined at activity level is always executed before the trigger procedure defined at workflow level.
- **Evaluation mode** The evaluation process allows defining what further activities are to be inactivated and started. This is called the “evaluation condition”. This can be done by invoking an external procedure or by using the link table. The link table allows the definition of logic between activities. The external procedures allow the coding of logic in an external application.
- **Evaluation procedure** Name of the Nirva procedure that will be ran when the evaluation mode has been set to “PROCEDURE”. The procedure receives the current activity status and is then responsible for feeding back to the workflow the list of the next activities to inactivate and start.
- **Join mode** Method for evaluation if an activity can be started or not. This parameter is used only when the link table is used to control the logic. The join mode can be set to “SCRIPT” or “PROCEDURE”. When an activity terminates, the workflow evaluates all the outgoing links to be activated (evaluation condition) and for each target activity, evaluates if it can start or not. The latter process is called “join condition”. By default, the join condition is simply a logical OR of all incoming links of the activity to check. Otherwise, the join condition can be evaluated by a script defined in the configuration of the activity (in perl-like language) or by a procedure in java or perl.
- **Join procedure** Name of the procedure to execute for the join condition when the join mode is “PROCEDURE”. If this information is empty and the join mode is “PROCEDURE”, then the join condition is a logical OR of the incoming links. This parameter is used only when the link table is used to control the logic.
- **Join script** Script to execute for the join condition when the join mode is “SCRIPT”. If this information is empty and the join mode is “SCRIPT”, then the join condition is a logical OR of the incoming links. This parameter is used only when the link table is used to control the logic. The join script is generally composed of a simple test between links. As example: “:A1 && :A2” is a join script telling to activate the activity only if both links from activities A1 and A2 are activated.
- **Autostart** Indicates if the activity must be started automatically. This is used to identify activities that can be started when a new process starts or the block activities to start when the block itself starts.
- **Time out** Time out value in seconds for a wait or receive activity to remain in RUNNING state and for a queue activity to remain in READY state. When this value is 0, no time out occurs.
- **Execution Time out** Time out value in seconds for a queue activity to remain in the RUNNING state. When this value is 0, no time out occurs.

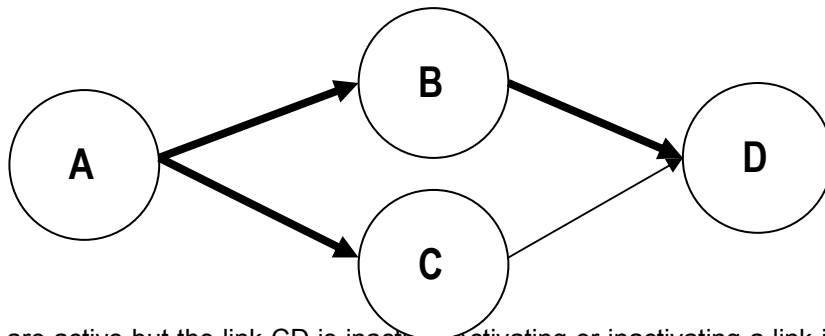
Link

A link connects two activities. An activity can have several incoming links and several outgoing links. For example:



The activity A has 2 outgoing links and the activity D has 2 incoming links.

A link receives a status of active or inactive:



The links AB, AC and BD are active but the link CD is inactive. Activating or inactivating a link is done when the activity terminates by a logic defined for each link in the configuration of the workflow. This mechanism is called “transition condition”. For example, in the previous figure, the transition condition of activity A has activated the AB and AC links.

The transition condition is defined by a Nirva procedure or by a script in perl-like language. If no procedure or no script is given, the default is to activate all outgoing links (even in case of error). The script for the transition condition can include process and activity variables.

When all outgoing links have been evaluated, the workflow will decide what activities to start next. For this purpose, the workflow takes all activated outgoing links of the current activity and evaluates its join condition. The join condition is defined at activity level (see Activity chapter). The default join condition is a logical OR between all incoming links.

In our previous example, if no join condition has been set for activity D, this one will start because the BD link is active. If the join condition is “C AND B”, then D will not be started because the link CD is inactive.

A link is defined using the following parameters:

- **Source** Source activity. A link of an activity inside a block cannot have its source outside this block.
- **Target** Target activity. A link of an activity inside a block cannot have its target outside this block.

- Transition mode Method for activating the link. The transition mode can be set to “SCRIPT” or “PROCEDURE”.
- Transition procedure Name of the procedure to execute for the transition condition when the transition mode is “PROCEDURE”. If this information is empty and the transition mode is “PROCEDURE”, then the link is activated as soon as the source activity terminates.
- Transition script Script to execute for the transition condition when the transition mode is “SCRIPT”. If this information is empty and the transition mode is “SCRIPT”, then the link is activated as soon as the source activity terminates.

The links are used only if the evaluation method defined at activity level is “LINK_TABLE”.

Variables

Variables can be defined at process and activity level. A variable is identified by its name and has a value.

Dedicated commands can define, set, remove and access these variables.

They can be also included in the transition condition and join condition scripts.

Variables are kept on disk in order to prevent loss in the case of unexpected system power off or crash.

The workflow provides standard system variables:

- NVW_STATUS Activity status (activity level).
- NVW_REASON Activity error reason (activity level).
- NVW_COUNT Activity instance counter for a single process (activity level).

All variables prefixed with “NVW_” are reserved and should not be written directly by the application.

Triggers

Some triggers can be defined when starting and stopping activities. For example, they can be used for sending information to an external audit or tracking system.

It is possible to define a trigger for all activities of a workflow. It is then defined at workflow level. There is a start and stop activity trigger.

It is also possible to define a specific trigger for each activity. There is a start and stop activity trigger.

The workflow level start trigger is executed before the activity level trigger.

The workflow level stop trigger is executed after the activity level trigger.

A trigger is a simple procedure called by the workflow. If the procedure does not exist, it is simply not executed and no error is generated.

The trigger procedures receive the parameters NVW_WORKFLOW (workflow name), NVW_ACTIVIY (activity name) and NVW_PID (Process Id). Stop trigger procedures receive the NVW_STATUS parameter that can take the values “COMPLETED”, “ERROR” or “TIMEOUT”.

Data

Each process receives its own space for storing data. This is supplied as a Nirva registry container (persistent container). In this way any kind of structured data can be associated to a process or activity.

For example this could be used to store any patient related data when the workflow is used in a medical application.

This data is kept with the process until the process is removed by the purge mechanism.

Some workflow specific and Nirva SYSTEM service commands allow access to this data.

Correlation tables

The correlation tables are used to associate a process with certain business data. For example in a banking application, the correlation tables allow retrieving processes to be associated with an account or customer number.

Several correlation tables can be defined in a workflow. A correlation table is identified by its name.

A correlation table is simply a hash table with pairs key/value. The key is the business data (ex account number or command number) and the value is the PID (process ID). Several PIDs can be associated to a given key.

The cross reference that lists all business keys associated to a given PID is stored in the system data associated to the process. Several business keys can be associated to a single PID.

The correlation tables are managed in a persistent way on disk.

Dedicated on-screen utilities in the Workflow cross reference between PIDs and correlation data.

Audit

All information about process operations is kept with the system data associated with each process.

Information can be accessed from dedicated on-screen utilities in the web interface. The audit data can only be accessed from the PID. Therefore, the PID must be known to access information related to it. This means it is not possible to display audit information of several processes in a list or to search inside audit data for several processes. However it is possible to use an external auditing system (from the database for example) by using activity and process level triggers.

Auditing information for a process is removed when the process is purged.

Engine

The workflow engine is Nirva itself. Nirva provides all necessary functionality to act as a workflow engine optionally on a distributed architecture. Different Nirva components can be involved in the workflow engine. The main ones are:

- **Listeners** The Nirva listeners are mainly used to process Queue workflow activities. The listener questions the workflow for activities to be processed, processes them and then reports the completion status (completed or error). A Nirva listener is a named object that can create several physical threads running in parallel. This allows improving the scalability of the system by allocating different system resources for different activities.
- **Scheduler** The scheduler can also act as a listener for processing certain Queue activities. The processing of these activities can be scheduled for given days and time but only one thread per activity is allowed. The scheduler can also be used to report events to the workflow. For example, a scheduled task can poll a directory for new files to arrive and then creates a new workflow process or reports the event to another workflow activity.
- **Connectors** The client connectors can be used by external systems or applications to communicate with the workflow and to optionally process activities. For example, an external program can report an event to a receive activity by sending a message to Nirva using the web service connector.
- **Web interface** A Nirva application with a web user interface can signal the workflow when an activity is completed. For example, the manual indexing of documents can be implemented in a Queue activity. Several staff acting as indexers can request from the workflow the next document to index (index activity), perform the task, signal the workflow when they have finished and then issue a new request to the workflow for the next available document to index. The workflow provides a locking mechanism for each queue activity insuring that not two indexers will receive the same document to index.

Processing

Creating a process

A process is created by sending the `PROCESS:CREATE` command to the workflow. The process is then created and optionally started if the option to start it immediately has been given.

All tables necessary to run the process are created in the process directory including the list of activities and links as they were defined in the workflow when the process was created. In this way, if the list of activities or links is modified, this will affect only processes created after this modification.

When a workflow is created all related activities and links receive the `INACTIVE` state.

Starting a process

A process is started by sending the `PROCESS:RUN` command to the workflow or by giving the option `START=YES` to the `PROCESS:CREATE` command.

A process can also be restarted by using the `PROCESS:RESET` that can be followed by a `PROCESS:RUN` command.

When a process is started, all activities with the autostart flag set are automatically started.

Ending a process

A process terminates when an activity of type `END` or `ERROR` without a parent is encountered in the process logic. The process status is then set to `COMPLETED` or `ERROR`.

A process can also be terminated by sending the `PROCESS:END` command. The process status is then set to `STOPPED`.

Process purge

Once a process has been set to `COMPLETED`, `ERROR` or `STOPPED` status, it is controlled by the retention mechanism. The retention mechanism keeps the process available in the system until the retention delay has expired. At this time, its status is changed to `DELETED` and the process is placed in a purge queue. This queue must be then called from a Nirva listener using the `PROCESS:PURGE` command. The deployment of the purge listener is under the responsibility of the application because the purge includes a call to triggers that must be run under the context of the application.

Process recovery

In the case of a system failure, processes that were running may have been accidentally stopped. The workflow service maintains a list of all processes currently running. When the workflow starts, this list is examined and the corresponding processes are transferred to a recovery list. These processes can then be recovered automatically by running the `PROCESS:RECOVERY` command from a dedicated listener or scheduled task. Typically this task should not be run often and a simple scheduler task running every hour should be enough to recover processes. It is also recommended to call `PROCESS:RECOVERY` from the application init procedure (the application executing workflow tasks).

Running activities

When starting the process, the activities marked with the autostart flag are started.

Running an activity can vary according to the kind of activity:

- An `Invoke` activity is started immediately and the system waits for it to terminate. The status is `RUNNING` while it runs and `COMPLETED` or `ERROR` when it ends. The activity procedure is executed. The activity procedure must return a session variable named `NVW_WORKFLOW_ERROR` that is not empty or generate a Nirva error in order to report a completion error. Otherwise, the execution is considered successful. The workflow procedure receives the following parameters: `NVW_WORKFLOW`, `NVW_PID` and `NVW_ACTIVITY` corresponding respectively to workflow name, process ID and activity name.
- A `Queue` activity has received `READY` state and has been placed in an activity queue. It can then be processed by a Nirva listener that will get it from the queue using the `ACTIVITY:GET` command. The

activity status is then set to RUNNING state. The listener must signal the end of the activity by sending the ACTIVITY:COMPLETE or ACTIVITY:ERROR commands setting the activity status to the appropriate state.

- A Receive activity is directly set to RUNNING state waiting for a signal from the application. The application must then send the ACTIVITY:COMPLETE or ACTIVITY:ERROR commands.
- Error and End activities do nothing at execution step so they are immediately set to COMPLETED state. If these activities are part of a block, the corresponding block activity is then set to COMPLETED state for an End activity and to ERROR state for an Error activity. If the activity is defined at process level, the process itself is set to COMPLETED or ERROR state.
- A block activity has no execution in itself but contains some activities marked with the autostart flag that the service runs when the block activity starts. The block activity is set to RUNNING state until an End or Error activity is reached inside the block.
- A wait activity is set to RUNNING state until the delay occurs. It is then placed in a time out queue to be processed by a dedicated listener using the ACTIVITY:TIMEOUT command.

When a process starts or when an asynchronous activity completes, the workflow first evaluates activities to start next and then proceeds to the inactivation of these activities, related links and target activities. Inactivating an activity means setting its status to INACTIVE state.

Managing time out and wait activities

A time out can be associated to Queue or Receive activities. A Wait activity is an activity that only has an associated time out. The Queue activity may have a second time out for the execution part of the activity (the first time out sets the time by which the activity is in the queue, ready to be executed).

Time outs are managed by internal tables and the application must define a time out listener or scheduler in order to process time outs. The reason is that when an activity goes into time out, its status is changing and the workflow must evaluate the activities to start next. This can only be done in the context of an application because the activity triggers are ran.

The time out listener or scheduler procedure will only have to call the ACTIVITY:TIME_OUT command. This command checks all activities that are in time out and sets them to time out status.

Scripts

Transition condition

The transition condition script is attached to each outgoing link. It decides whether the link must be activated or not. The transition script is written in perl code that is evaluated inside an "if" perl test. If the result of this test is true, then the link is activated.

The script can contain workflow variable names. The variable names are prefixed with a \$ character, followed by the activity name, followed by a ":" character and then by the variable name that should not contain spaces or special characters. If this is a process level variable, the activity name is let blank and the

“:” character is omitted. If the activity name is blank but the “:” character is given, the variable refers to the current activity (Source activity of the link). For example: \$COMPOSE:DOCNAME is a variable named DOCNAME of the COMPOSE activity, \$GSTATUS is a process variable named GSTATUS, \$:TEST is a variable named TEST for the current activity.

Variables are evaluated before running the script. For example, if the script is "\$:NVW_STATUS" eq "COMPLETED" this will be send as if("COMPLETED" eq "COMPLETED") if the status of the current activity is COMPLETED.

When the transition condition script is not given, the link is activated when the activity terminates disregarding the actual termination status.

Join condition

The join condition script is attached to each activity having incoming links. It decides whether the activity can be started or not. The join script is written in perl code that is evaluated inside an “if” perl test. If the result of this test is true, then the activity can start.

The script can contain workflow variable names like in the transition script but generally it will only test the incoming link status. A link status is evaluated by giving the source activity name with a “:” character before.

For example, if the activity A3 has 2 incoming links from A1 and A2 activities, the script “:A1 && :A2” will authorize to run the activity A3 only if the links from respectively A1 and A2 are activated.

By default, if no join condition script is given, the workflow simply calculates a logical OR between all incoming links.

Installation

The workflow service is delivered as a NIRVA package and can be installed like any NIRVA service directly from the NIRVA configuration web site. Please see the NIRVA configuration chapter in the NIRVA user's guide for further information.

The workflow deployment can be done using the workflow service configuration import and export functions. Since the processing of workflow activities is under the responsibility of applications, the applications must also be installed.

The workflow may use Nirva listeners and scheduled tasks for processing some activities. The applications running the workflow service commands provide them. There are particularly three tasks that these applications must supply:

- A time out processing listener
- A recovery scheduled task
- A purge listener or scheduled task

Time out

The time out listener is a simple Nirva listener (see Nirva documentation) that calls the `ACTIVITY:TIME_OUT` command. The listener can be set with a sleep time of one second and following the amount of time out processes to manage, one or several threads can be set for the listener.

Recovery

The recovery is a simple scheduled task running each hour and calling the command `PROCESS:RECOVERY`.

In fact the recovery is only used in case of power off or any other accidental crash on the system.

Purge

The purge mechanism can be a listener or a scheduled task (running during the night for example) that calls the `PROCESS:PURGE` command. If the workflow uses many processes and there is a purge trigger

procedure defined, it may be preferable to use a listener instead of a scheduled task and to reserve several threads for this listener. The listener sleep time can be set to one second.

If the workflow does not define a retention delay, the purge mechanism is not necessary.

Configuration

The workflow service configuration is entirely dynamic and available from a web browser.

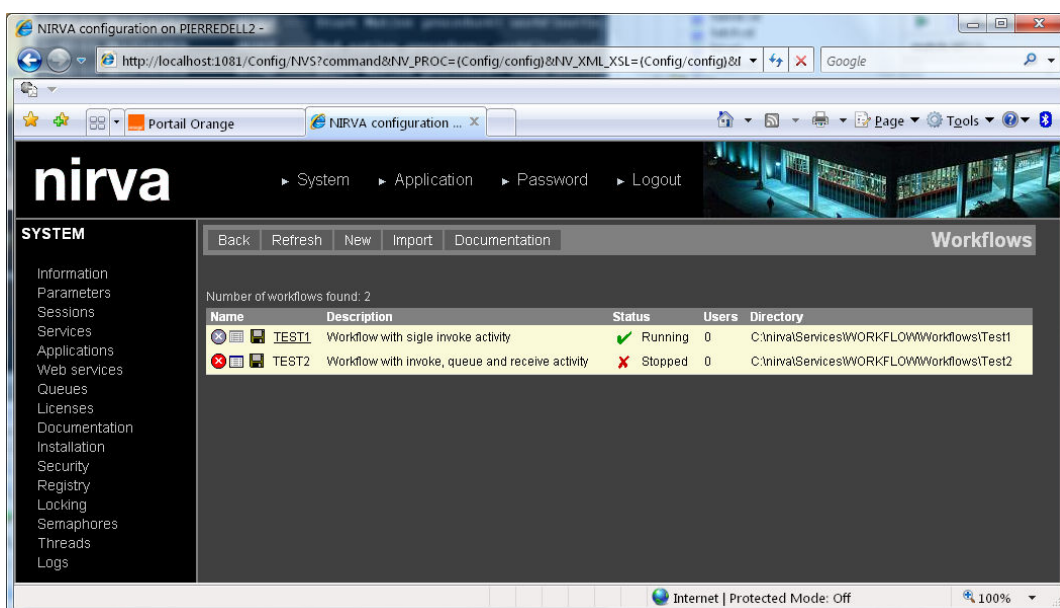
The configuration of the workflow service is accessible directly from the main list of services of the NIRVA configuration web site.

Some of the configuration screens or actions can be restricted by permissions. In order to access all screens the user should have the ADMIN and WORK workflow service permissions. See the Main Nirva documentation for further information about security and user permissions.

Workflow

Workflow list

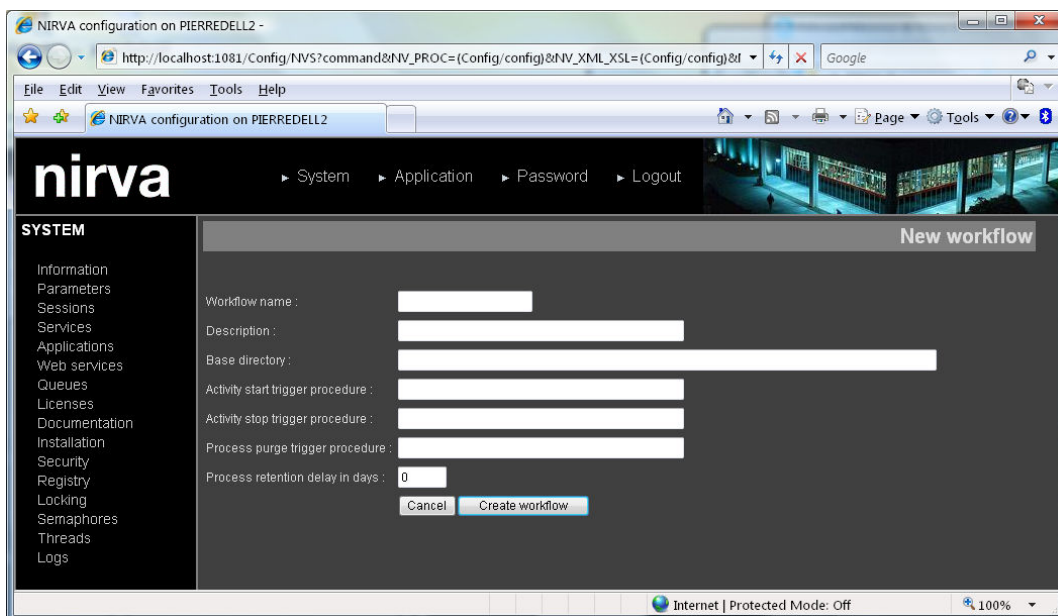
When entering the workflow service configuration, Nirva displays the list of available workflows:



Name is the workflow name, “status” indicates whether the workflow is running or not, “users” gives the number of users currently connected to this workflow and “directory” is the workflow directory where all workflow data is stored.

Creating a new workflow



To create a new workflow, press the “New” button from the workflow list window. This displays the following screen:




Description of the workflow parameters:

Workflow name	Uniquely identifies the workflow. Cannot contain spaces or special characters. Mandatory parameter.
Description	Workflow description.
Base directory	Directory for the workflow data. If the given directory does not exist, the command creates it.
Activity start trigger procedure	Name of a trigger procedure that will run each time an activity starts. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide. If this parameter is blank, no start trigger procedure will be launched.
Activity stop trigger procedure	Name of a trigger procedure that will run each time an activity ends. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide. If this parameter is blank, no stop trigger procedure will be launched.
Process purge trigger procedure	Name of a trigger procedure that will run each time a process is removed from the system. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide. If this parameter is blank, no stop trigger procedure will be launched.
Retention delay	Retention delay in days for the processes of the workflow. After the retention delay, a process is removed from the system by the purge mechanism. If this value is 0, the process is removed as soon as it ends.

Starting and stopping a workflow

For starting or stopping a workflow, press the  or  buttons near the workflow status from the workflow list screen.


Graphically configuring a workflow

For accessing the graphical workflow configuration, press the  button from the workflow list screen. The graphical configuration is described in the chapter "[Graphical configuration](#)".

Removing a workflow

For removing a workflow, press the  button from the workflow list screen.

Exporting a workflow

For exporting workflow definition in XML format, press the  button from the workflow list screen.

Importing a workflow


To import a workflow definition previously exported in XML data, press the "Export" button from the workflow list window. This displays the following screen:



The workflow name is inside the XML data but may be changed thus allowing a copy of an existing workflow with a different name. If the workflow does not exist, it is created. If it exists, the workflow definition is changed according to the imported definition file.

If the workflow does not exist, the base directory for storing workflow data must be given.

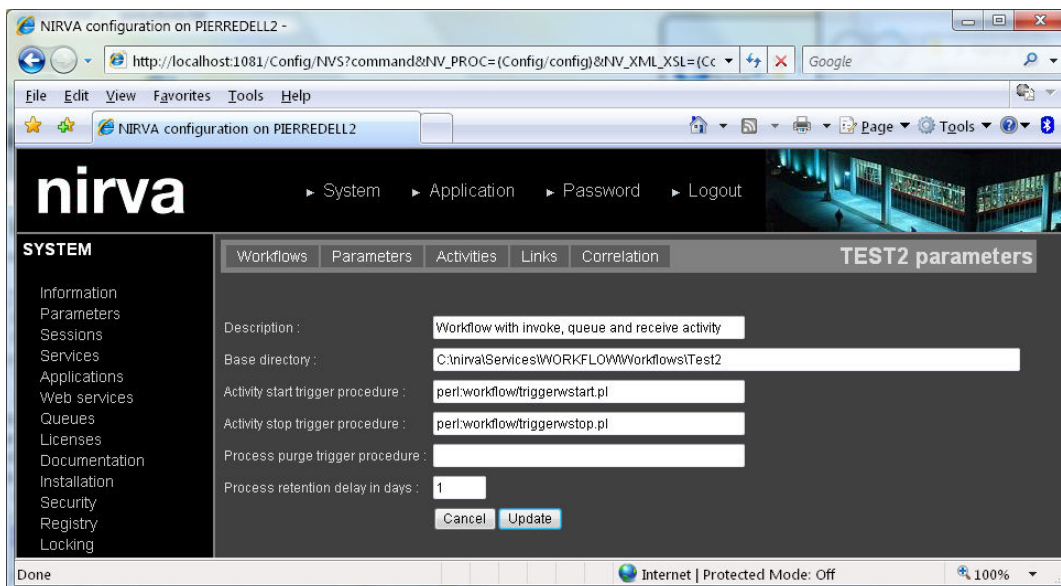
Modifying workflow parameters

For modifying the workflow parameters, press the  button near the workflow name from the workflow list window. This action is possible only when the workflow is not running (stopped).

The following parameters can be modified:

- General workflow parameters
- Activities
- Links
- Correlation tables

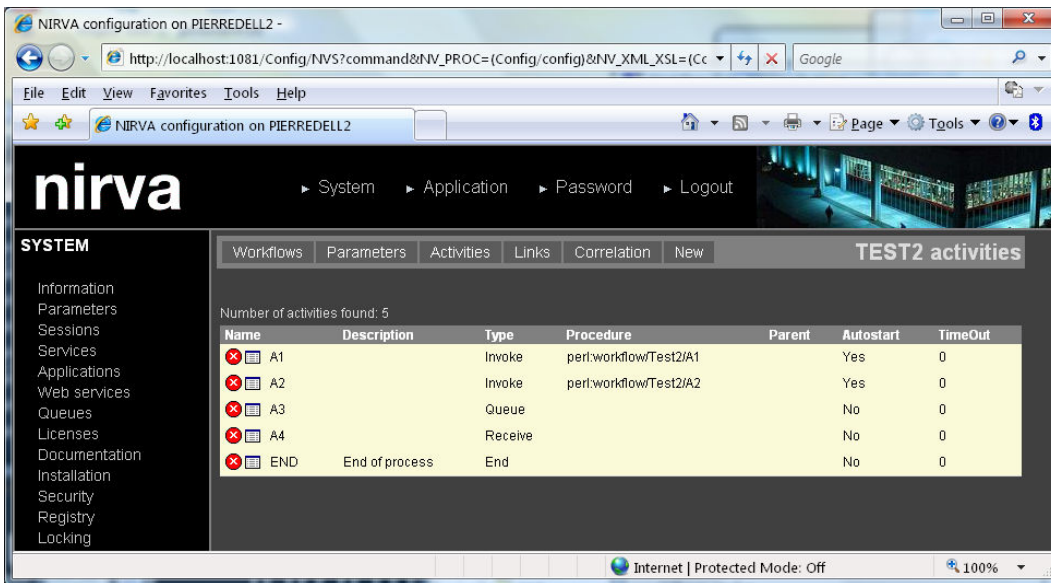
General workflow parameters:



Please see "[Creating a new workflow](#)" for a description of the workflow parameters.

Activities

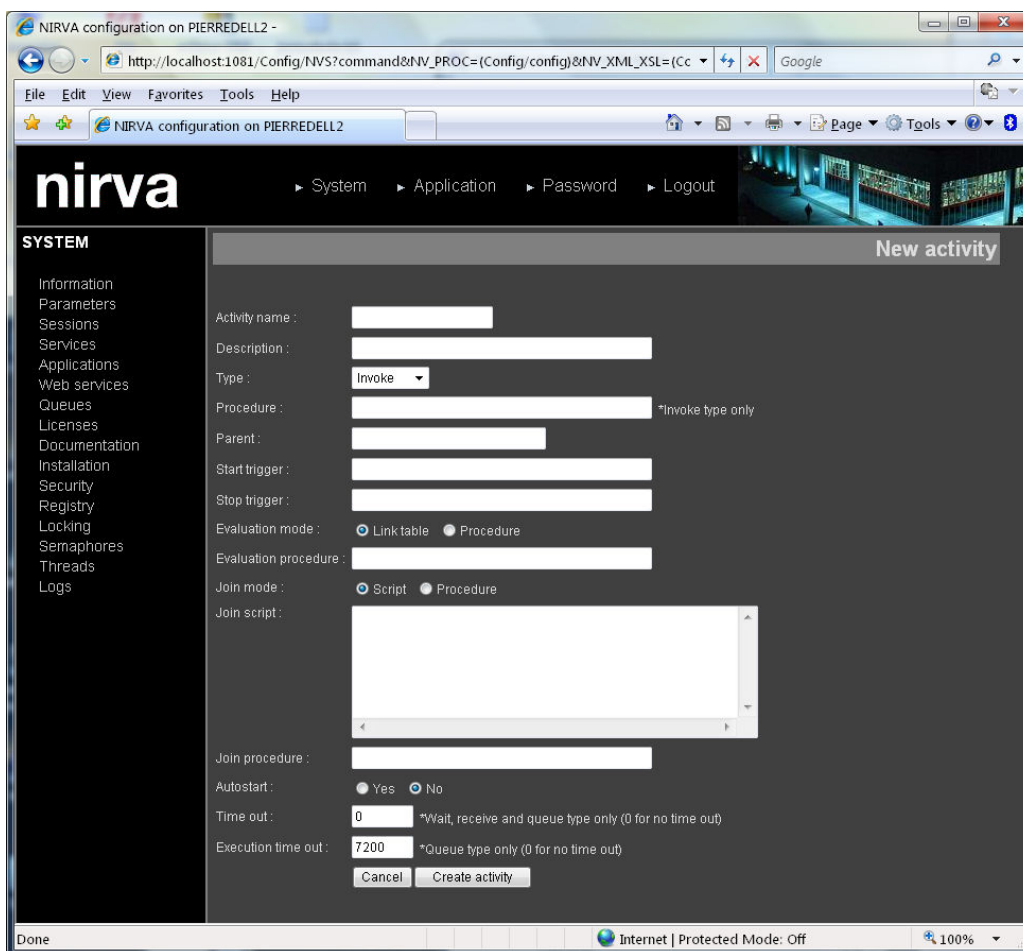
This screen defines the activities of the workflow. If the activity list is modified, this impacts only newly created processes. Nirva always stores a process with the activities it had at creation time.



“Name” is the activity name, “Type” is the activity type (Invoke, Queue, Receive, Wait, End, Error or Block), Procedure is the procedure name for invoke activities, “Parent” is the block parent activity, “Autostart” is a flag indicating whether the activity should automatically starts when the process starts and “TimeOut” is the activity time out (for Queue, Receive and Wait procedures).

Creating an activity

For creating an activity, press the “New” button from the activity list:




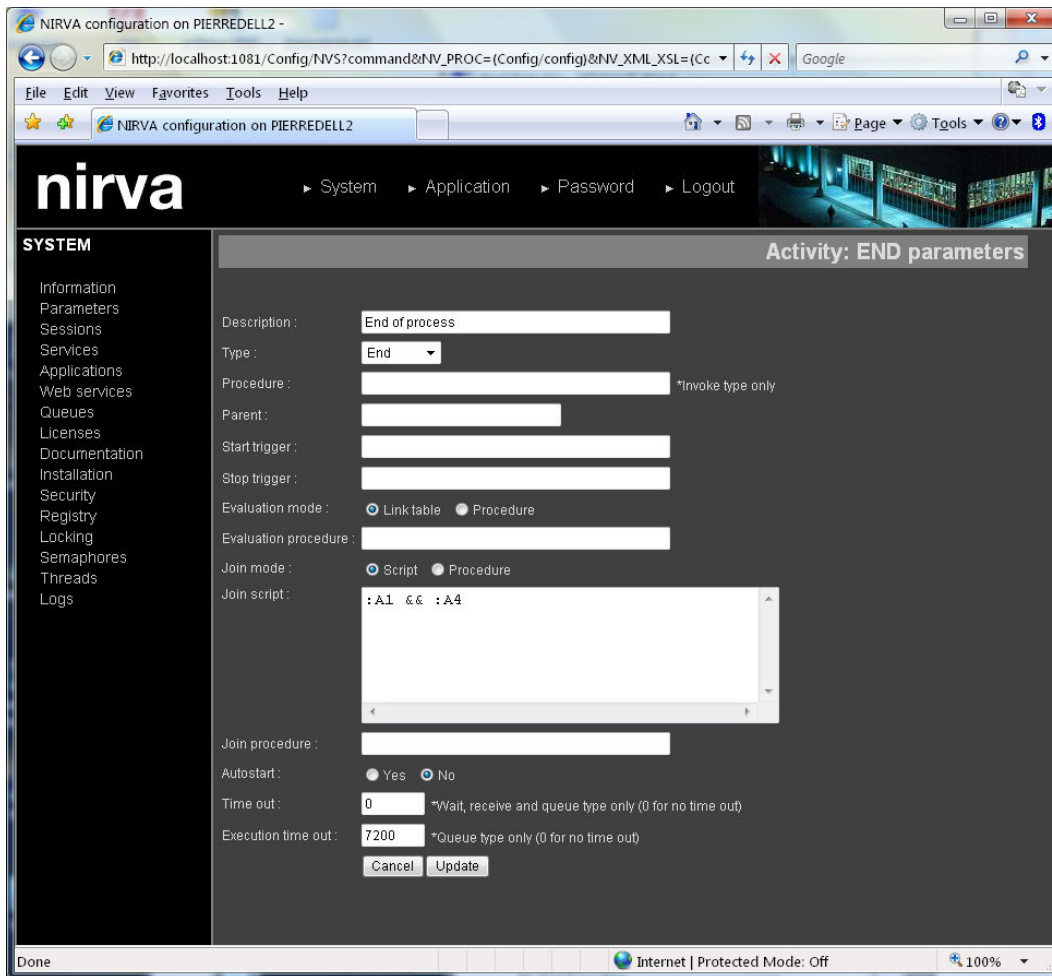
Description of the activity parameters:

- Name** Activity name. Uniquely identifies the activity. Should not contain any space or special characters.
- Description** Activity description.
- Type** Activity type. See the chapter [“Activity”](#) for further information about activity types.
- Procedure** Procedure for an Invoke activity. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide.
- Parent** Name of the parent activity for an activity inside a Block activity. This parameter allows defining all the activities inside a Block.
- Start trigger** Name of a Trigger procedure ran each time the activity is started. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide.
- Stop trigger** Name of a Trigger procedure ran each time the activity ends. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide.
- Evaluation mode** Evaluation mode for the next activities to start. This can be “Link table” for using the link table or “Procedure” for using an external procedure.

<i>Evaluation procedure</i>	Evaluation procedure when the evaluation mode has been set to "Procedure". The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
<i>Join mode</i>	Join mode. Can be set to "Script" for using a script or to "Procedure" for using an external procedure.
<i>Join procedure</i>	Join procedure when the Join mode has been set to "Procedure". The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide. The join procedure receives 3 parameters named NVW_WORKFLOW, NVW_PID and NVW_ACTIVITY corresponding to workflow name, process ID and activity name. It must set a session variable named "NVW_RESULT" that can have the values "TRUE" or "FALSE" following the result of the join condition.
<i>Join script</i>	Join script when the Join mode has been set to "Script". The join script is a test that is automatically evaluated by the workflow. It must be written in perl. The activity and process variables can be used but generally the join script will use only link variables. Link variables are boolean variables having the format :SOURCE_ACTIVITY where SOURCE_ACTIVITY is the name of the source activity. For example ":A1 && :A2" is a valid script that will validate the join condition only if the links from activity A1 and A2 are validated. See the "Scripts" chapter for further information.
<i>Autostart</i>	Flag telling if the activity must be started when starting the process or the block. If this parameter is set to "Yes" and the activity has a parent (part of a block activity), it will be started automatically when the parent activity starts. If this parameter is set to "Yes" and the activity has no parent, it will be started automatically when the process starts
<i>Time out</i>	Time out value in seconds for a Wait or Receive activity to remain in RUNNING state and for a queue activity to remain in READY state. When this value is 0, no time out occurs.
<i>Execution time out</i>	Time out value in seconds for a Queue activity to remain in RUNNING state. When this value is 0, no time out occurs.


Modifying an activity

For modifying an activity, press the  button from the activity list:



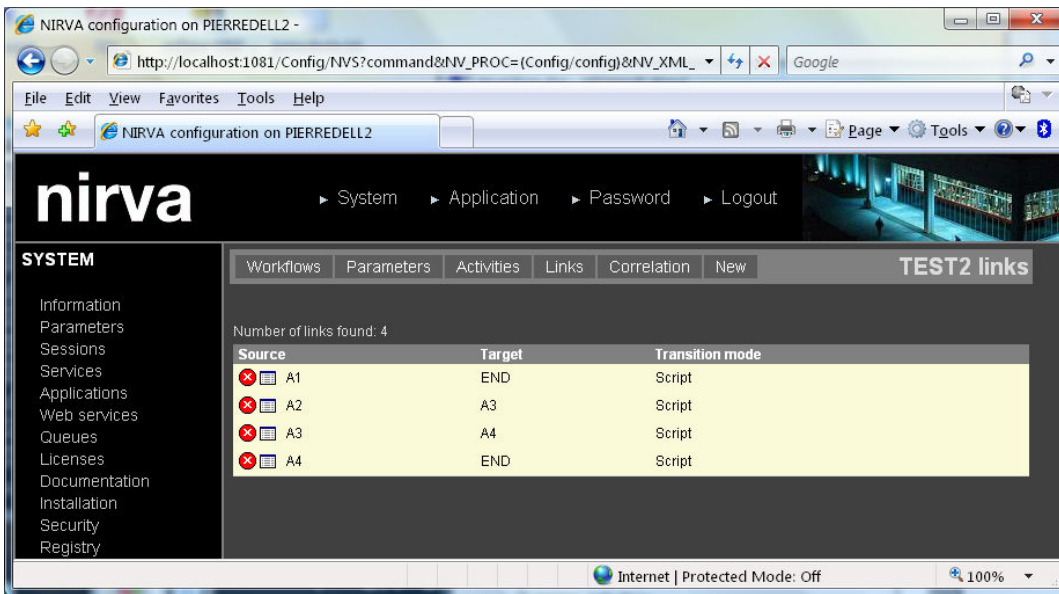
See the previous chapter “Creating an activity” for a description of activity parameters.

Removing an activity

For removing an activity, press the  button from the activity list.

Links

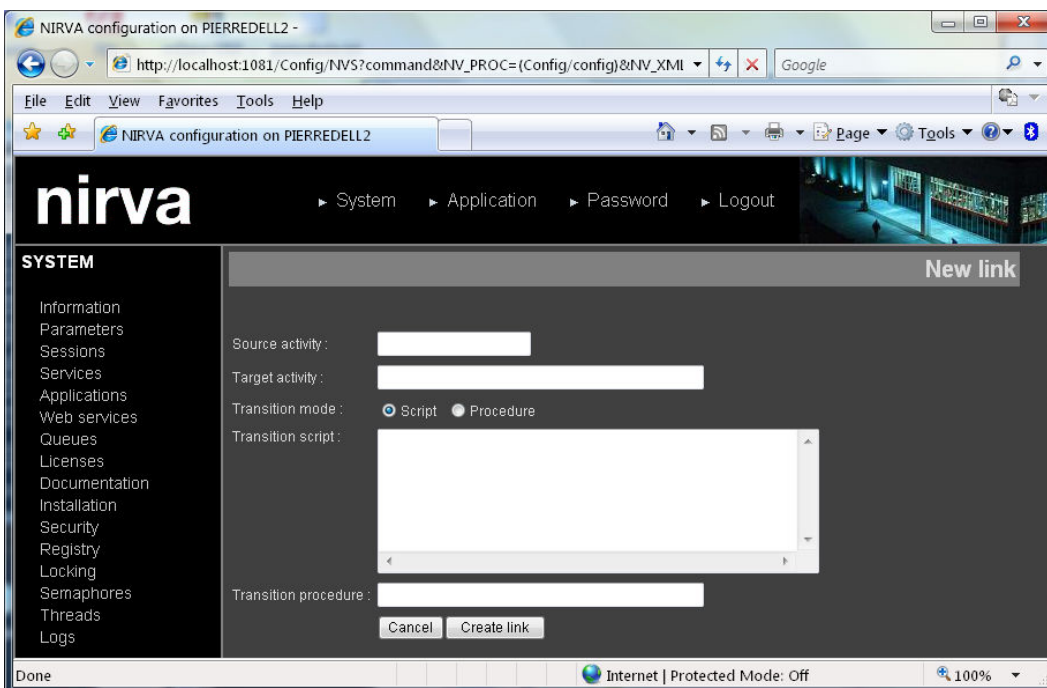
This screen allows defining the link between activities when the evaluation mode of the activity has been set to “Link table”. See the chapter “[Link](#)” for further information about links.



“Source” is the source activity name, “Target” is the target activity name and transition mode defines the way the link is activated (by script or by procedure).

Creating a link

For creating a link, press the “New” button from the link list:



Here is the description of the link parameters:

Source activity Source activity name.

Target activity Target activity name.

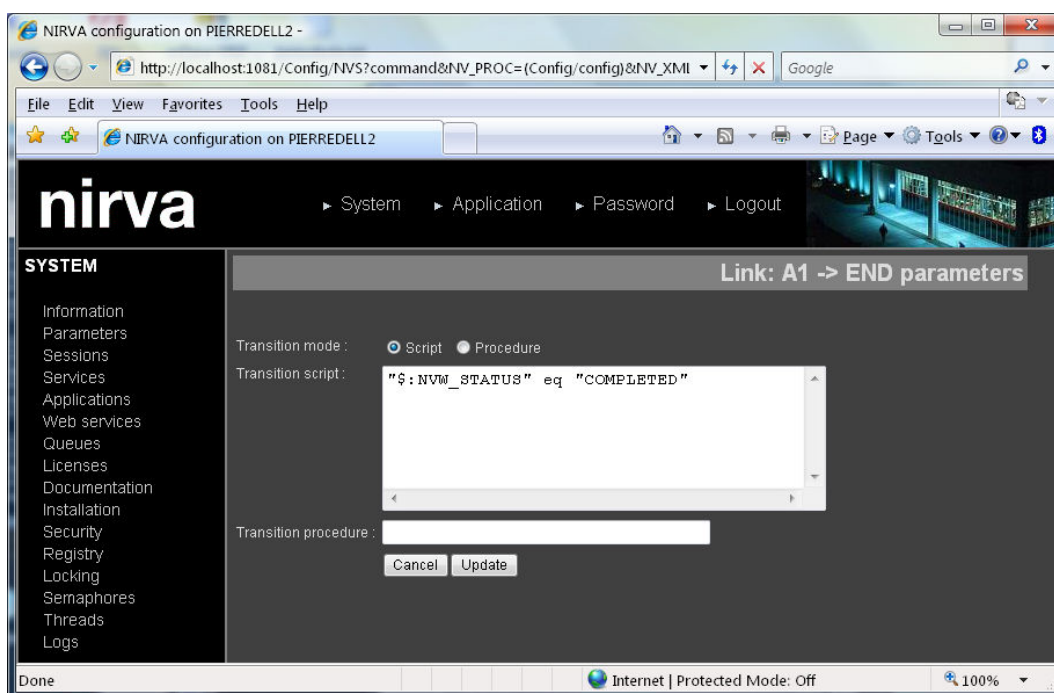
Transition mode Transition mode. Can be set to “Script” for using a script or to “Procedure” for using an external procedure.

Transition script Transition script when the transition mode has been set to “Script”. The transition script is a test that is automatically evaluated by the workflow. It must be written in perl. The activity and process variables can be used. An activity variable has the format \$ACTIVITY:VARIABLE where ACTIVITY is the activity name and VARIABLE is the variable name (ex.: \$A1:VAR1). If the activity name is not given (but the : character is given), this refers to the source activity (ex.: \$:VAR1). A process variable has the format \$VARNAME (ex.: \$VAR1). See the chapter “[Scripts](#)” for further information.

Transition procedure Transition procedure when the transition mode has been set to “Procedure”. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide. The join procedure receives 4 parameters named NVW_WORKFLOW, NVW_PID, NVW_SOURCE, NVW_TARGET corresponding to workflow name, process ID, activity source and target names. It must set a session variable named “NVW_RESULT” that can be set to the values “TRUE” or “FALSE” following the result of the transition condition.

Modifying a link

For modifying a link, press the button from the link list:



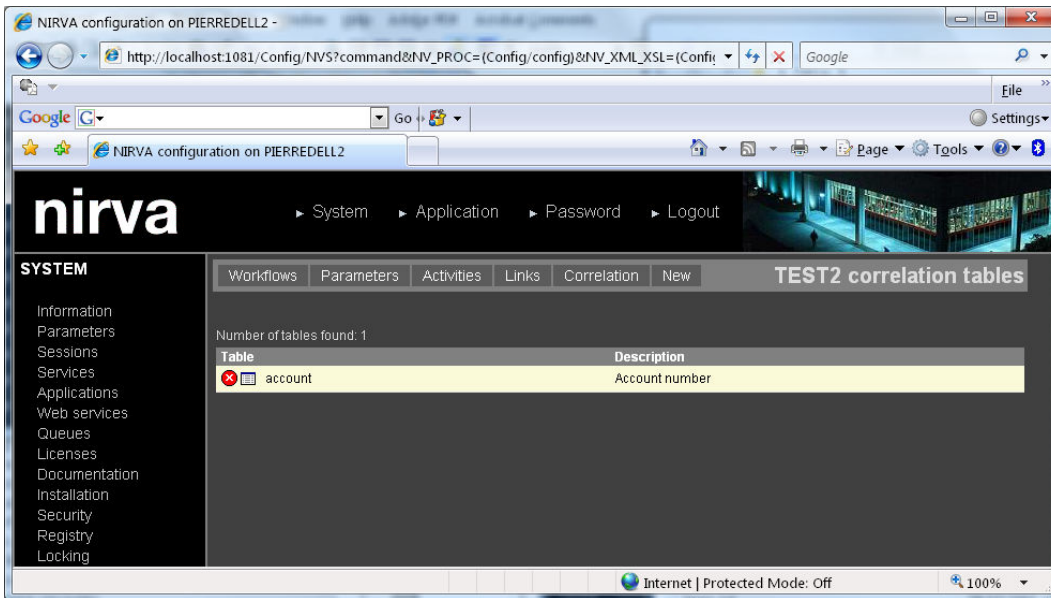
See the previous chapter “Creating a link” for a description of link parameters.

Removing a link

For removing a link, press the button from the link list.

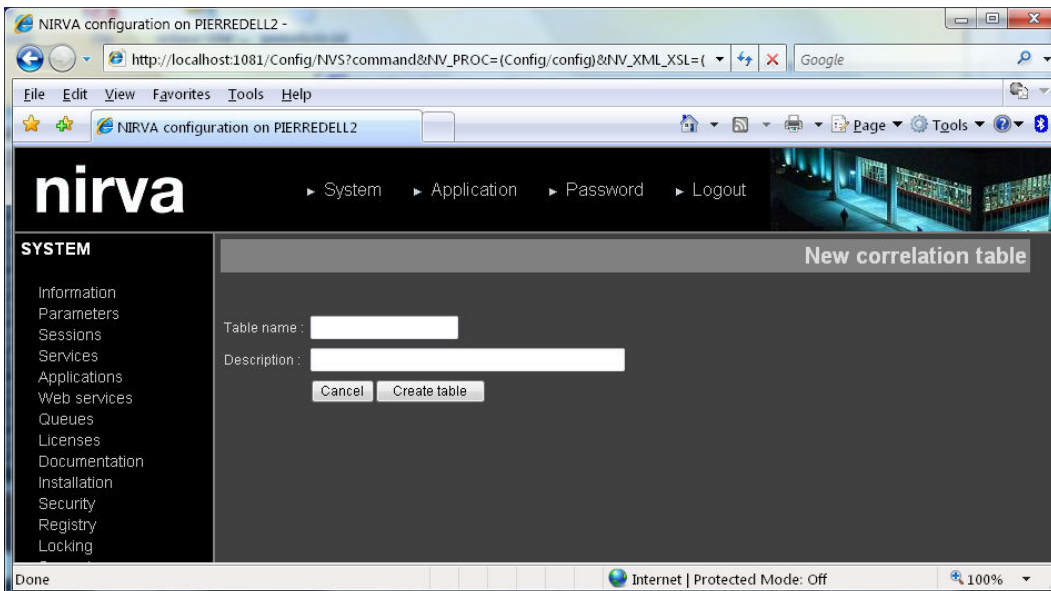
Correlation tables

This screen allows defining the list of correlation tables for the workflow. Correlation tables are used to associate business data to workflow processes. See the chapter [“Correlation tables”](#) for further information about correlation tables.




Creating a correlation table

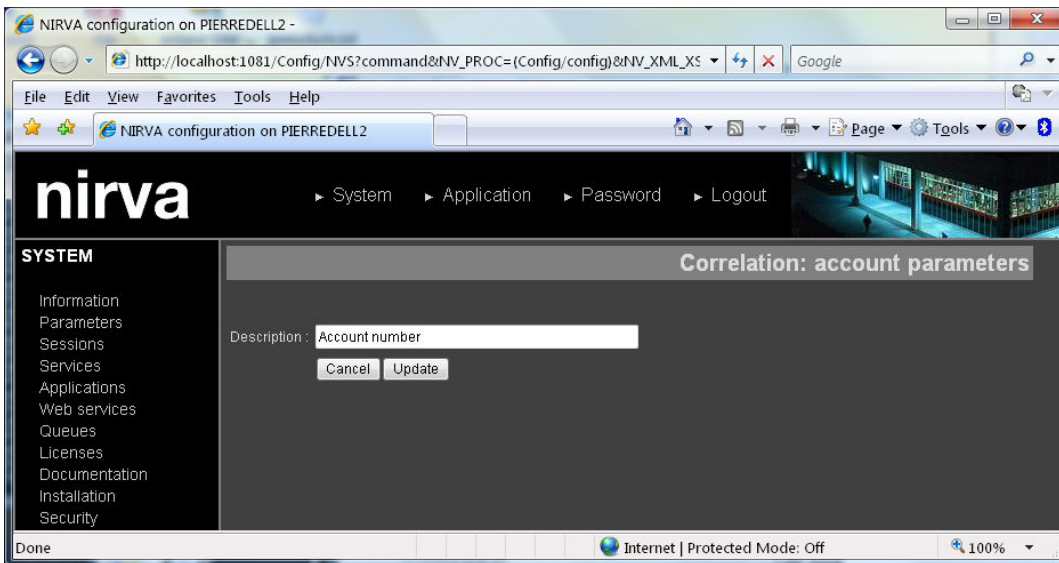
For creating a correlation table, press the “New” button from the correlation table list:




The correlation table name should not contain any space or special character. The correlation table name is not case sensitive.

Modifying a correlation table

For modifying a correlation table, press the  button from the correlation table list:



Removing a correlation table

For removing a correlation table, press the  button from the correlation table list.

Production

Search

When the workflow is started, clicking on the workflow name displays the production search screen. The list of processes can be searched with three criteria:

- By date (creation date of the process)
- By process ID (PID)
- By correlation tables

By date

The default mode is to search by date (creation date of the process):

The screenshot shows the Nirva configuration web interface. The browser address bar displays the URL: `http://localhost:1081/Config/NVS?command&NV_PROC=(Config/config)&NV_XML_XSL=(Config/config)&NV_...`. The interface includes a navigation menu with options like System, Application, Password, and Logout. A sidebar on the left lists various system components such as Information, Parameters, Sessions, Services, Applications, Web services, Queues, Licenses, Documentation, Installation, Security, Registry, Locking, Semaphores, Threads, and Logs. The main content area is titled "TEST2 processes" and displays a table of active processes. The search criteria are set to "By: Date", "From:" (empty), "To:" (empty), "Status: Active", and "Sort: Newest first". The table shows 14 processes found, with the following data:

Pid	Global status	Date
376F3853B5FB97011FC59185F62950EB	Active	2008-07-25 10:19:51
50D21057A9CC61EFC3998F6758657FF	Active	2008-06-27 17:34:18
D223F74C54FB5481ED4602C19E7D7C8C	Active	2008-06-27 17:15:32
C7239CA887D25D05F9C9898B8F1FD293	Active	2008-06-27 17:12:38
ABC937D189F1CCCA4A9D32E3A8184074	Active	2008-06-27 16:43:51
648D62ED7A2F2BDF4588B9DC2740D065	Active	2008-06-04 15:03:17
15702244802532A04FEE19E788F1D2A2	Active	2008-06-04 15:03:11
DC9C6ACB0D14B1DD20FC149983999951	Active	2008-06-04 15:03:11
28A684F445DFFABDF1E658F583D9770D	Active	2008-06-04 15:03:11
601BE38825605D3AF668EB05196A96FF	Active	2006-11-28 13:00:15

The search criteria are:

From

From date/time criteria. The general format of the DATE is YYYY-MM-DD HH:MM:SS where DD is the day from 1 to 31, MM is the month from 1 to 12, YYYY is the complete year, HH the hour, MM the minutes and SS the seconds. The time part is not mandatory.

The From criteria accepts another format that is a relative date before the actual date. The format is then the following: $-Xy$ where X is an integer and y a letter that can take the value d for days, h for hours, m for minutes and s for seconds. For example $-2h$ will be the actual time minus 2 hours.

To

To date/time criteria. The general format of the DATE is YYYY-MM-DD HH:MM:SS where DD is the day from 1 to 31, MM is the month from 1 to 12, YYYY is the complete year, HH the hour, MM the minutes and SS the seconds. The time part is not mandatory.

The To criteria accepts another format that is a relative date before the actual date. The format is then the following: $-Xy$ where X is an integer and y a letter that can take the value d for days, h for hours, m for minutes and s for seconds. For example $-2h$ will be the actual time minus 2 hours.

Status

This can restrict the search to active or terminated process. By default only the active processes are retrieved. The active or terminated status is the global PID status. A terminated process is candidate to the purge mechanism and stays in the workflow until its retention delay expires. Since workflow service version 3.06 one can search for process terminated with success or error and for stopped processes. If the process has been created with a workflow service version prior to 3.06, these extended searches will return empty results.

By process ID

If the process ID is known, one can search directly for it:

The screenshot shows the Nirva web interface in a browser window. The search is performed by Process ID. The search criteria are: Search By: Process ID, Pid: 648D62ED7A2F2BDF4588B9DC2740D065. The results show 1 process found.

Pid	Global status	Date
648D62ED7A2F2BDF4588B9DC2740D065	Active	2008-06-04 15:03:17

By correlation table

This is a convenient way to retrieve a process according to business data:

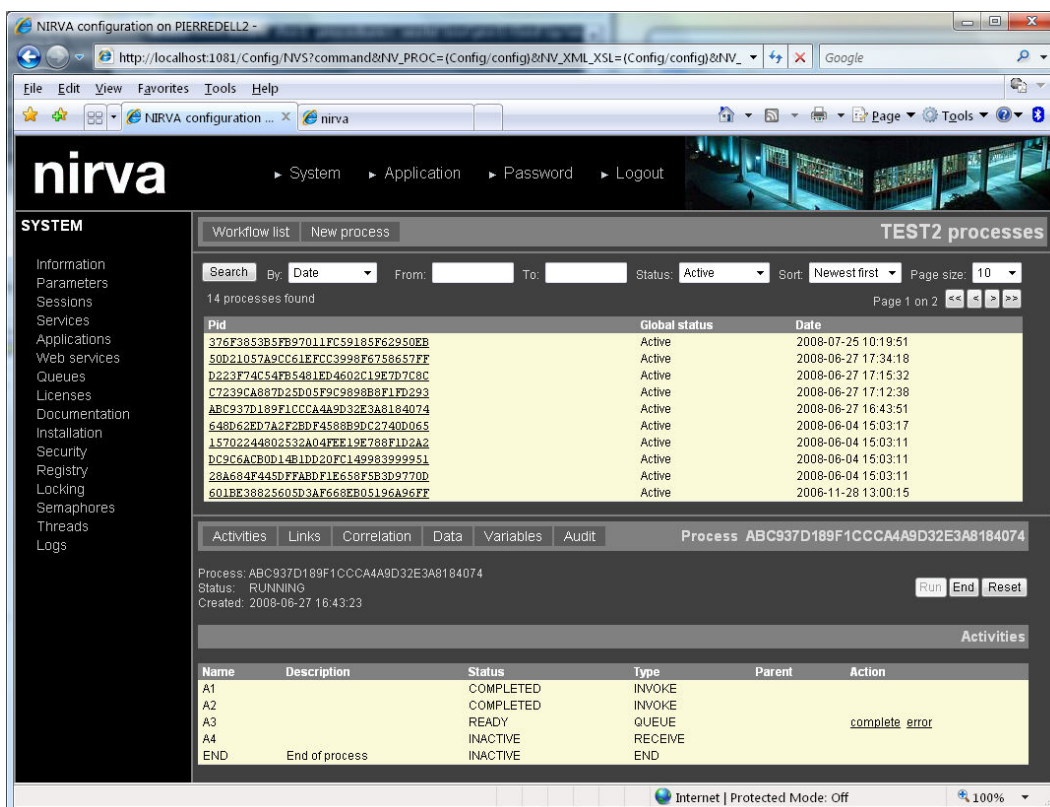
The screenshot shows the Nirva web interface with a search by correlation table. The search criteria are: Search By: Correlation, Table: account - Account number, Value: 123, Sort: Newest first. The results show 2 processes found.

Pid	Global status	Date
648D62ED7A2F2BDF4588B9DC2740D065	Active	2008-06-04 15:03:17
15702244802532A04FFE19E788F1D2A2	Active	2008-06-04 15:03:11

The available correlation tables are the ones defined at workflow level (see the [Correlation tables configuration](#)).

Detail process data

From any of the search screen, clicking the process ID shows the detail process in the bottom part of the screen:



It is possible to view detailed process related information including activities, links correlation values, data associated to the process, variables and audit data. It is also possible to perform certain actions on the process or activities but this should be reserved for maintenance or test purposes. If this is done, the user must be connected to the application that normally processes the workflow activities.

The first part of the screen displays the process ID, the process status, the creation date and the termination date. Following the process status it is possible to start, end or reset the process.

The second part of the screen displays one of the following information:

- Activities
- Links
- Correlation data
- Associated data
- Variables
- Audit data

Activities

Name	Description	Status	Type	Parent	Action
A1		COMPLETED	INVOKE		
A2		COMPLETED	INVOKE		
A3		READY	QUEUE		complete error
A4		INACTIVE	RECEIVE		
END	End of process	INACTIVE	END		

This is the list of the process activities with their actual status. It is possible to manually change some of these values by using the links displayed in the Action column. In any case this should be reserved for test or maintenance purpose since the normal workflow actions should be made by dedicated applications or external programs. If these links are used, the user must be connected to the application that normally processes the workflow activities because changing an activity status may start automatically some triggers or others activities.

Links

Links		
Source	Target	Active
A1	END	YES
A2	A3	YES
A3	A4	NO
A4	END	NO

This just displays the actual link table associated to the process if there is one.

Correlation data

New correlation value		
Table	Value	Action
account	123	remove

This is the list of correlation data associated to the process. It is possible to add or remove certain values.

Associated data

Refresh New key New entry Export Import

Name	Type
input	<input checked="" type="checkbox"/> container
lists	<input checked="" type="checkbox"/> container
output	<input checked="" type="checkbox"/> container
parameters	<input checked="" type="checkbox"/> table

Modify parameters

NAME	DESCRIPTION	VALUE
SOURCE	Source type	DIRECTORY
COLOR	Background color of the document	BLUE
SIZE	Size of the output document	13456

This screen works like the standard Nirva registry editor but allows accessing the registry dedicated to each process where one can store any process data. Please see the main Nirva documentation (configuration chapter) for information about using the Nirva registry editor.

Note: in this screen, in order to save screen space, the process information is not displayed.

Variables

Name	Value	Activity
NVW_STATUS	RUNNING	
NVW_COUNT	1	A1
NVW_REASON		A1
NVW_STATUS	COMPLETED	A1
NVW_COUNT	1	A2
NVW_REASON		A2
NVW_STATUS	COMPLETED	A2
NVW_COUNT	0	A3
NVW_REASON		A3
NVW_STATUS	READY	A3
NVW_COUNT	0	A4
NVW_REASON		A4
NVW_STATUS	INACTIVE	A4
NVW_COUNT	0	END
NVW_REASON		END
NVW_STATUS	INACTIVE	END

This screen displays all process variables at process or activity levels. The variables starting with “NVW_” are system variables created and maintained by the workflow service. It is possible to edit or add variables (except for system variables).

Auditing

Date	Activity	Action	Info
2008-06-27 16:40:09		Process start	
2008-06-27 16:41:03	A1	Activity run	occurrence = 1
2008-06-27 16:41:44	A1	Activity complete	
2008-06-27 16:41:48	A2	Activity run	occurrence = 1
2008-06-27 16:41:48	A2	Activity complete	
2008-06-27 16:41:49	A3	Activity ready	

Each process action is registered and can be displayed using the audit screen. This is an internal workflow auditing system. Any other auditing can also take place in the workflow by the way of the activity triggers.

Creating a new process

A new process can be created using the button New process at the top of the production screen:



This should be reserved for test and maintenance purpose since process creation is generally made by a Nirva or an external application.

Once the process is created, it is immediately displayed in the bottom part of the screen (process detail) and can get started (Run button). The user must be connected to the application that normally creates processes before doing so. Starting a process may invoke activities and triggers that should normally run in the context of an application.

Graphical configuration

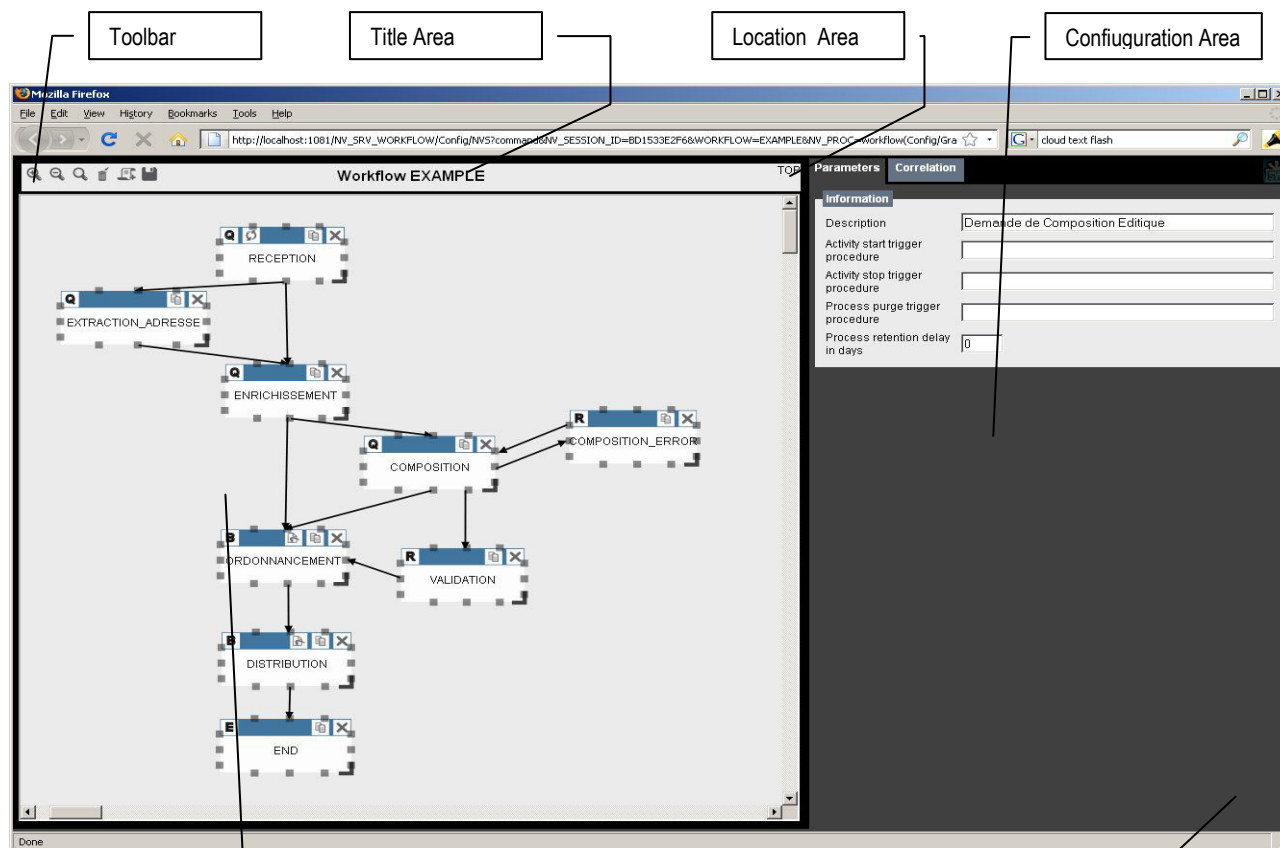
You access the graphical configuration by pressing the  button from the workflow list screen.

The graphical configuration uses SVG technology. This technology is natively available on Firefox browsers but requires a plug-in installation on Internet explorer. The workflow service directly embeds the plug-in for Internet explorer and automatically installs it on your workstation if necessary. In case of installation problem, you can proceed with a manual installation by executing the file SVGView.exe found on the subdirectory Services/WORKFLOW/Wroot/Config/Graphics of your Nirva directory. This file must be copied on the client computer and executed in order to proceed to the manual installation.

The plug-in installation and usage may display some internet security warning or require autorisation on your browser. In some cases, after the installation of the plugin, it is required to reload the page to take the plugin into account.

The graphical interface has been successfully tested on Firefox 3.0 and Internet explorer 7 and 8.

Interface organization



The graphical editor is divided into different zones:

Drawing Area

Message Area

Toolbar

A toolbar zone allowing general operations on the workflow or drawing area: zooming, delete the selection, create an activity and save the workflow. The components of the toolbar are detailed below.

Title area

The title area recalls the workflow which is being edited. The name shown is the one which has been selected when the edition started.

Drawing area

The drawing area gives the graphical representation of active block of the workflow being edited. It allows moving activities, drawing links, zooming into a part of the drawing, etc. The components of the drawing area are detailed below.

Location area

The location area gives the path to the active block (i.e. the block which is currently being edited). A click on the names of any of the ancestor blocks allows a quick access to that block. A click on the “TOP” link allows returning to the workflow’s main block.

Configuration area

The configuration area allows to view/change the settings of the global workflow parameters, the correlation tables associated to the workflow, and when applicable the parameters of the currently selected activity or link. The components of the configuration area are described below.

Message area

The graphical workflow has a non intrusive message area where messages may appear. For example, when saving the workflow, a confirmation that the workflow has been correctly saved will appear in this area.

Information messages appear in green as in the example below:







Error messages appear in red as in the example below:



In both cases, messages are kept for a limited amount of time (10 seconds). A click on the cross on the upper right corner will immediately close the message box. A click anywhere else in the message box will keep it until it is manually closed.

Toolbar

The toolbar presents five buttons which are always visible:

Icon	Action
	Zoom in the drawing area
	Zoom out the drawing area
	Restore the previous zoom configuration. In particular after an area-zoom.
	Delete the currently selected link or activity

Icon	Action
	Add an new activity
	Save the workflow. The modifications made to the workflow are saved to the given workflow. The workflow must exist and not be active when saving. After saving the workflow, a verification of its consistency is done and any errors will appear in the message area.

The following icons may also be visible when appropriate:

Icon	Action
	Zoom out of the currently active block (make the parent block the active one).

Configuration

Workflow parameters

The “Parameters” tab of the configuration area allows setting the general workflow parameters.

The screenshot shows the configuration interface with the following details:

- Parameters** (selected) | Correlation
- Information**
 - Description:
 - Activity start trigger procedure:
 - Activity stop trigger procedure:
 - Process purge trigger procedure:
 - Process retention delay in days:
- Drawing area**
 - Width:
 - Height:

Information group

The content of the information group of this tab contains the following inputs:

<i>Description</i>	Allows setting the description of the workflow
<i>Activity start trigger proc.</i>	Allows setting the workflow's global activity start procedure called each time an activity of the workflow is started.
<i>Activity stop trigger proc.</i>	Allows setting the workflow's global activity stop procedure called each time an activity of the workflow is stopped.
<i>Process purge trigger proc.</i>	Allows setting the workflow's process purge trigger procedure called each time a process is purged.
<i>Process retention delay</i>	Allows setting the number of days a process is kept before being allowed to be purged.

Drawing area group

The drawing area group allows settings the parameters of the drawing area for the workflow. The parameters which can be set are the following:

<i>Drawing width</i>	The width of the drawing area.
<i>Drawing height</i>	The height of the drawing area.





Attention: using large drawing areas may introduce a slow zoom with internet explorer.

Correlations

The "Correlations" tab of the configuration area allows adding, editing and removing correlation tables to the workflow. The figure below shows the content of this area



First are listed the available correlations which have two action icons to the right. A click on  deletes the corresponding entry. A click on  enables the edition of the corresponding entry.

Next a blank entry allows adding a new correlation entry by clicking on .

Activity parameters

When an activity is selected, a new tab appears in the configuration area. This tab contains the currently selected activity's parameters. Depending on the type of activity, some of the parameters may be disabled.

Parameters **Correlation** **Activity**

Information

Name

Description

Type

Parent

Autostart Yes No

Procedure

Triggers

Start trigger

Stop trigger

Evaluation

Link table Procedure

Procedure

Join

Join mode Script Procedure

Join script

Join procedure

Time outs

Time out *Wait, receive and queue type only (0 for no time out)

Execution time out *Queue type only (0 for no time out)

Information group

<i>Name</i>	Allows setting the name of the activity. This name must only contain alpha numerical characters and cannot contain spaces (any spaces will be removed). It must be unique throughout the workflow independently of blocks (i.e. even two activities of different blocks may not have the same name).
<i>Description</i>	Allows setting the description of the activity
<i>Type</i>	Allows setting the type of the activity.
<i>Parent</i>	Shows the name of the parent block for this activity. This value cannot be changed directly.
<i>Autostart</i>	Allows setting the autostart indicator for the activity.

Procedure group

This group allows setting the procedure to be run by the workflow in case of an activity of type invoke.

Triggers group

<i>Start trigger</i>	Allows setting the activity's start trigger procedure.
<i>Stop trigger</i>	Allows setting the activity's stop trigger procedure.

Evaluation group

This group allows setting the evaluation mode which determines how to evaluate which activities need to be activated when the activity comes to an end.

<i>Link table</i>	When active the evaluation mode is to use the link table. In this case, the links of the workflow are used to determine which activities should be enabled.
<i>Procedure (button)</i>	When active the evaluation mode is to use a procedure.
<i>Procedure (input)</i>	Allows setting the name of the procedure to be called when the evaluation mode is set to procedure.

Join group

This group allows setting the join mode which determines when this activity is to be activated.

<i>Join mode</i>	Can be either script or procedure. In script mode a script is used to determine when the activity should be activated. In procedure mode a procedure is used to determine when the activity is to be activated.
<i>Script</i>	Allows setting the script to be used in "script" join mode.

Procedure Allows setting the name of the procedure to use in “procedure” join mode.

Time outs group

This group allows setting the time outs for the activity.

Time out Allows setting the timeout delay (in seconds) for activities of type Wait, Receive, and Queue

Execution time out Allows setting the execution timeout delay (in seconds) for activities of type Queue.

Link parameters

When a link is selected an extra tab appears in the configuration area. This tab allows viewing and setting the different parameters of the link.

The screenshot shows a configuration window with three tabs: Parameters, Correlation, and Link. The Link tab is active. It contains two main sections: Information and Transition. The Information section has two text input fields: Source (containing 'SORT') and Target (containing 'DISTRIBUTION'). The Transition section has a radio button group for Mode, with 'Script' selected and 'Procedure' unselected. Below the Mode group is a large empty text area for the Script. At the bottom of the Transition section is a text input field for Procedure.

Information group

Source Recalls the name of the source activity for the link.

Target Recalls the name of the target activity for the link

Transition group

Mode Allows setting the transition mode for this link. The transition mode can be either “script” or “procedure”. In script mode a script is used to determine if

the link should be activated when the source activity ends. In procedure mode a procedure is used instead.

Script The script to be used in script transition mode.

Procedure The name of the procedure to be called in procedure transition mode.

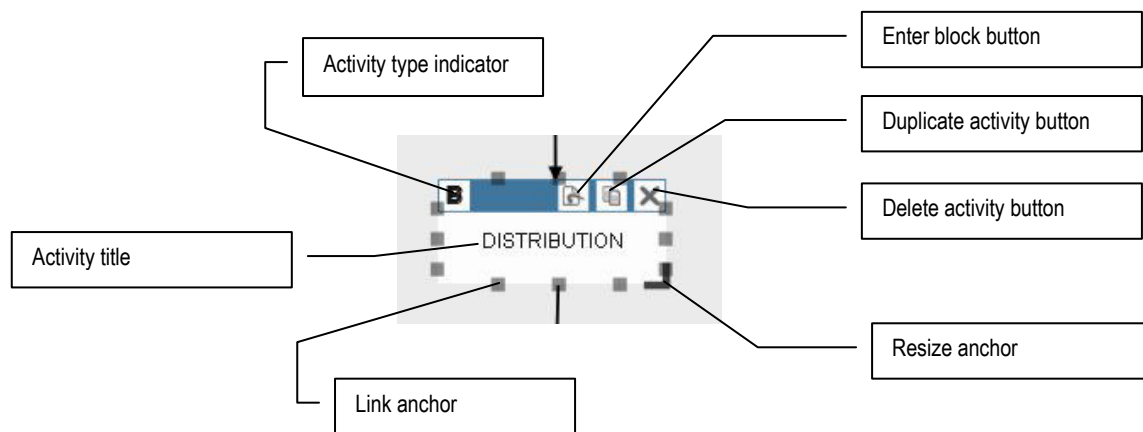
Drawing area

The drawing area allows different actions on activities (select, move, resize, delete and duplicate) and links (select, create, change anchors and delete).

At any time there may be zero or one active selection. If the selection is a link, it appears in red. If the selection is an activity

Activities

In the drawing area activities appear as show below:



Autostart indicator

Beside the activity type indicator the icon may appear. It indicates that the activity will start automatically when the process or parent block is started.




Activity type indicators

The activity type indicator shows the type of the activity. The table below gives the correspondences between the icons and the activity type.

Icon	Activity type
I	Invoke
Q	Queue
W	Wait
B	Block
E	End
Err	Error

Action icons

A series of action icons appear on the right of the activity menu bar. These icons are described in the following table.

Icon	Action
	Appears only on block activities and allows zooming into the block.
	Creates a copy of the current activity. All the parameters are duplicated except the activity name which is calculated to be unique throughout the workflow.
	Deletes the activity. In the case of a block activity, a confirmation message is shown indicating that all the descendant activities will also be deleted.

Resize anchor



The resize anchor allows changing the graphical size of the current activity. This allows organizing the workflow diagram favoring its readability.

Moving activities

An activity can be moved simply by dragging the activity. Any zone of the activity which is not otherwise associated to an action starts a drag.


Adding activity

An activity can be created by:

- Cloning another activity by clicking on its clone button () in its upper right corner.
- Clicking on the “add activity” button () in the tool bar.

Deleting activities

An activity can be deleted by:

- Selecting the activity and clicking on the “trash can” button () in the toolbar
- Clicking on the cross in its upper right corner
- Selecting the activity and pressing the “Del” key

Links

Adding links


A new link can be added by simply starting a new drag from a link anchor of the source activity and dropping the end point of the link on an anchor of target activity. If the link

Moving links


Both endpoints of a link may be moved by selecting the link and starting a drag from one of the anchors of the source or target activity link anchors.

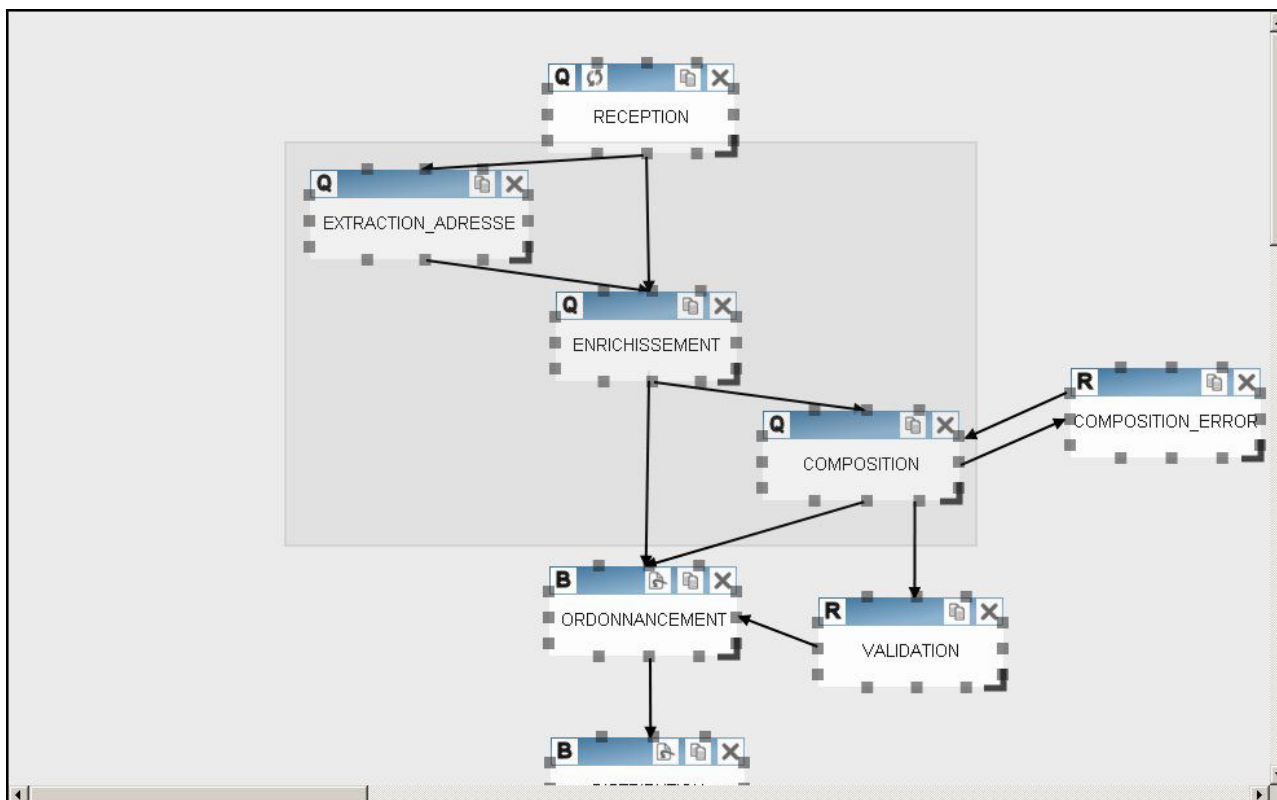
Deleting links

A link can be deleted by:

- Selecting the link and clicking on the “trash can” () button in the toolbar
- Selecting the link and pressing the “Del” key
- Dragging and releasing an endpoint of the link out of any activity link anchor

Area Zoom

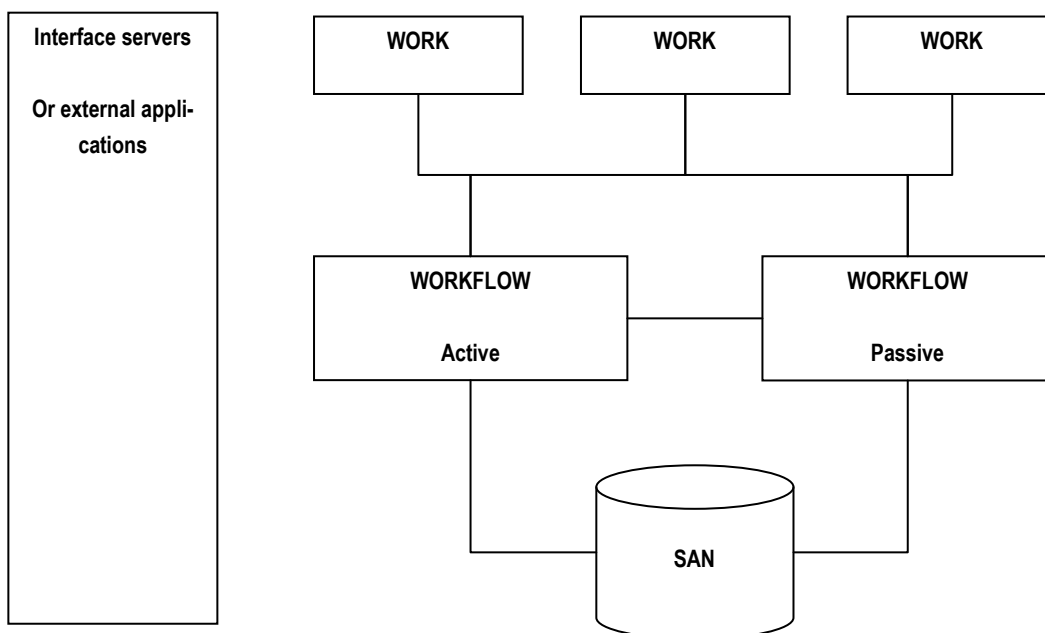
The editor allows zooming by selecting a target area. The user selects the area to zoom into using the mouse. The following screenshot shows a user selecting an area in which he would like to zoom into. It is possible to come back to the original zoom by clicking on the restore zoom button ().



Architecture

The workflow service and the application(s) using it can all run on a single server but in most cases a business solution will run on several computers in order to improve scalability and performance.

Here is a typical architecture:



Note: In a Nirva distributed architecture, all Nirva servers must be configured to use long session IDs. This guarantees that the session IDs are always unique across servers.

Workflow servers

The Workflow service is installed on a central computer that runs with a second computer in failover mode. The passive Workflow computer permanently checks if the active computer is up and running and will start when it fails.

Nirva directly manages the failover. To activate it, run Nirva with the option “-l peer address”. See the chapter Using Nirva / Starting and stopping Nirva server / Cluster mode in Nirva documentation for further information.

Each Workflow server must be the cluster of the other one in order to invert the role active/passive in case of real failover situation.

The Workflow server runs:

- The workflow service itself
- All synchronous activities.
- The workflow recovery, time out and purge tasks (always in the application context)

All must run in the context of a Nirva application.

SAN

A SAN file system is used to store both the Workflow data and the registry of Workflow servers. For each Workflow server, you should change the Nirva configuration (right after initial installation) to point the registry to a common SAN directory. This avoids synchronisation of Nirva parameters when a failover situation occurs.

In the same way, the Workflow data must be configured to a common SAN directory so that when a failover situation occurs, the new active server can directly retrieve the workflow data.

For this configuration to work properly, it is mandatory that both servers never work together in active mode.

Instead of using a SAN, it is possible to use NAS technology or simply a shared directory but the workflow requires many I/O to support crash condition and the performance will be better with a SAN.

Work servers

The work servers implement most of the workflow asynchronous tasks (generally queue tasks) in the context of the application. They can be specialised on some dedicated activities or not. Some of them can be in failover mode but in most cases they will be mounted on load balancing.

They communicate to the Workflow servers using Nirva intercommunication features. If the Workflow server is inside a failover cluster, the virtual address of the cluster must be used. This virtual address can be configured in the Nirva system parameters of the work servers.

Here is a typical example of code of a listener executing a queue activity on the work server:

```
# Open the connection to the workflow server
# WAPP is the name of the application running the workflow on the workflow server.
# WSERVER is the virtual address of the cluster workflow server (this virtual address
must be entered in Nirva configuration system parameters)
# WUSER is the WAPP user name (can be omitted if default user)
```

```

# WPASSWORD is the password
# The user must have enough rights to work on the workflow

# Open a connection to workflow server
NV::Command("NV_CMD=REQUEST:OPEN| NAME=|WORKFLOW| APPLICATION=|WAPP| SERVER=|(WSERVER)|
USER=|WUSER| PASSWORD=|WPASSWORD|");

while(1)
{
# Asks the workflow for a process with an activity to run
NV::Command("NV_REQUEST=|WORKFLOW| NV_CMD=|WORKFLOW:ACTIVITY:GET|
WORKFLOW=|MYWORKFLOW| ACTIVITY=|MYACTIVITY|");

NV::Command("NV_REQUEST=|WORKFLOW| NV_CMD=|OBJECT:GET| NAME=|PID|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID|");
$PID = $NV::RESULT;

# Leave if no process available
if($PID eq ""){last;}

# execute the activity here
...

# Set the activity to success (or error)
NV::Command("NV_REQUEST=|WORKFLOW| NV_CMD=|WORKFLOW:ACTIVITY:COMPLETE|
WORKFLOW=|MYWORKFLOW| ACTIVITY=|MYACTIVITY| PID=|$PID|");

# Stop if there is a global stop request
NV::Command("NV_CMD=|SESSION:CHECK_CLOSE_REQUEST|");
if($NV::RESULT eq "YES"){last;}
}

# Close the connection to workflow server
NV::Command("NV_CMD=REQUEST:CLOSE| NAME=|WORKFLOW|");

```

Note: the work servers are not necessarily Nirva servers. If they are not they can use one of the Nirva connectors to communicate with the workflow but they must manage themselves the virtual address for accessing the workflow servers mounted as failover cluster.

Interface servers

The architecture may also include interface servers. They are Nirva servers implementing some user interface or receiving external orders via client connectors (ex web service consumer, MQ Series listening, etc...).

They are generally mounted in load balancing. The load balancing is controlled by the external application communicating with the interface servers or by a hardware switch (synchronised on the Nirva ID session). For the MQ connector, there is no need of load balancing because this is automatically controlled by interrogating the MQ queues.

External applications

The external applications communicate directly with Workflow servers or across the interface servers.

They are not necessarily Nirva applications. For example, a java application on a J2EE environment can work with a Nirva workflow.

If they are not running on Nirva servers, they use one of the Nirva client connectors and they must themselves manage the access to the virtual address of a workflow server mounted in failover cluster.

Example

This chapter gives a simple example of a workflow.

This sample workflow depicts a supplier invoice validation application.

Description

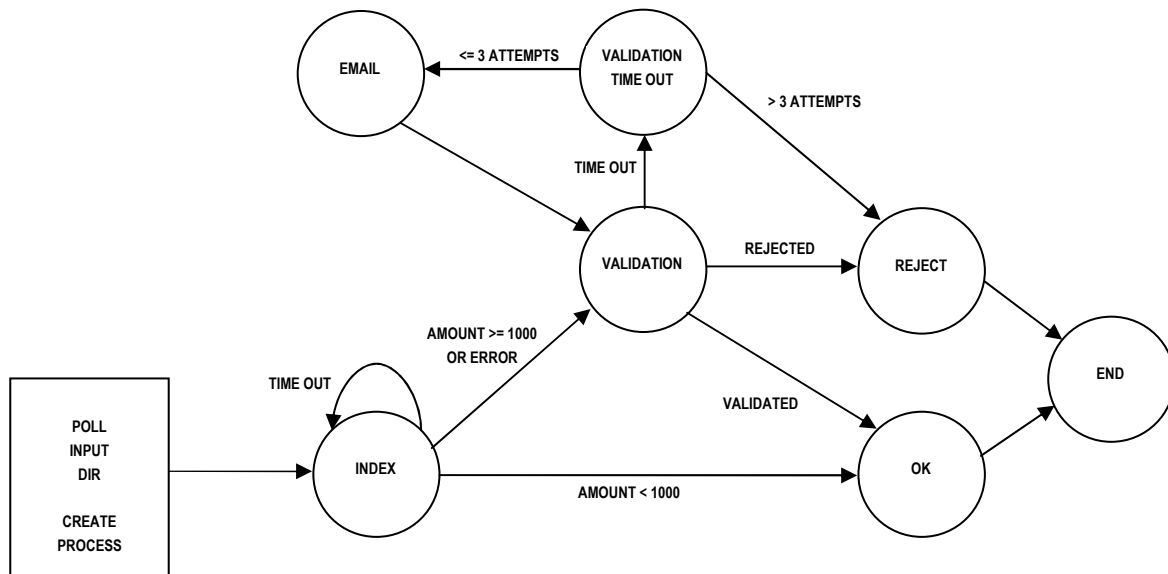
A listener checks an input directory for pdf files containing scanned invoices. Dedicated staff index these invoices by entering the supplier name and the amount. If the amount is greater than 1000 or if the indexer decides it, the invoice has to be validated by the supervisor. Otherwise the invoice is automatically validated.

For a manual validation the supervisor can validate or reject (with a reason) the invoice.

The supervisor has a certain time to do the validation (60 minutes for the purpose of the example but can be configured). If this is not completed in due time, an email is sent as a reminder of pending invoices validation (email emulation in the example). After three attempts (so three time outs) the invoice is automatically rejected with a time out reason.


In any case, the invoice is sent to an output directory (one for validated invoices and one for rejected invoices) with an xml file giving the index and the optional rejection reason.

Here is the workflow diagram:



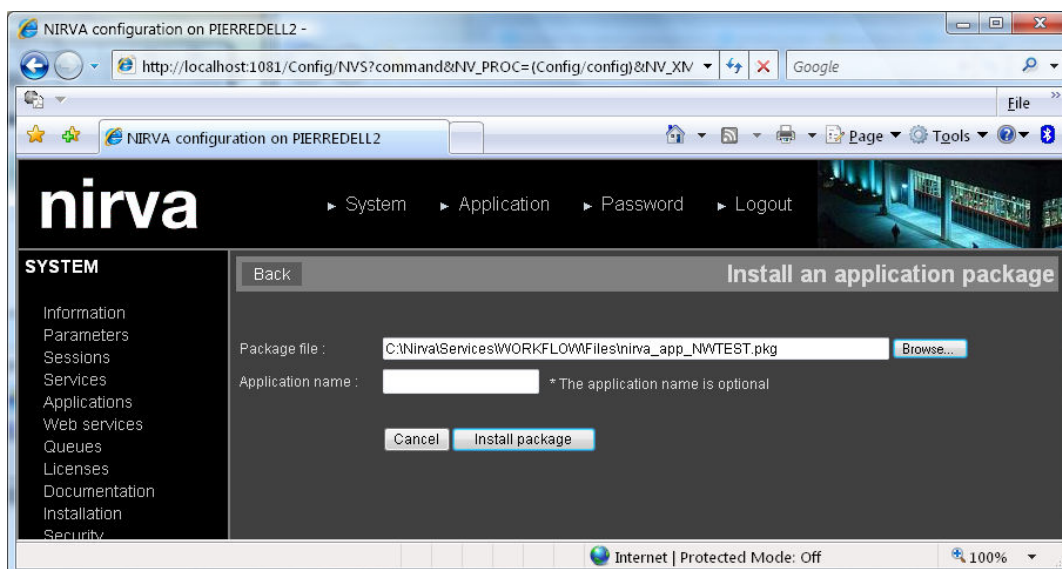
Installation

Prerequisites

- Nirva version 2.6.015 or greater. Nirva must be started. It is preferable to run Nirva in console mode (nvs -c) because the workflow example displays some log information in the console.
- Use Nirva configuration tool with nvadmin user or with a user having all necessary rights. In order to view workflow definitions, connect the Nirva configuration tool to the NWTEST application once installed.
- Workflow service version 1.00 or greater. The workflow service must be started. In order to start it, open the Nirva configuration tool, go to the System/Services menu and click the  button on the WORKFLOW service line.
- A license for running the both Nirva and the workflow service. Please contact Nirva systems to obtain test licenses.
- PDF acrobat viewer with Internet Explorer or Firefox plug-in.

Installation


To install the application, open the Nirva configuration tool and go to the System/Applications menu. Then press the “Install” button:



The application is named NWTEST (you do not need to give the application a name because it is already defined in the application package).

The application package file is located in the Files directory of the workflow service (typically "C:\Nirva\Services\WORKFLOW\Files\nirva_app_NWTEST.pkg").

Starting the application

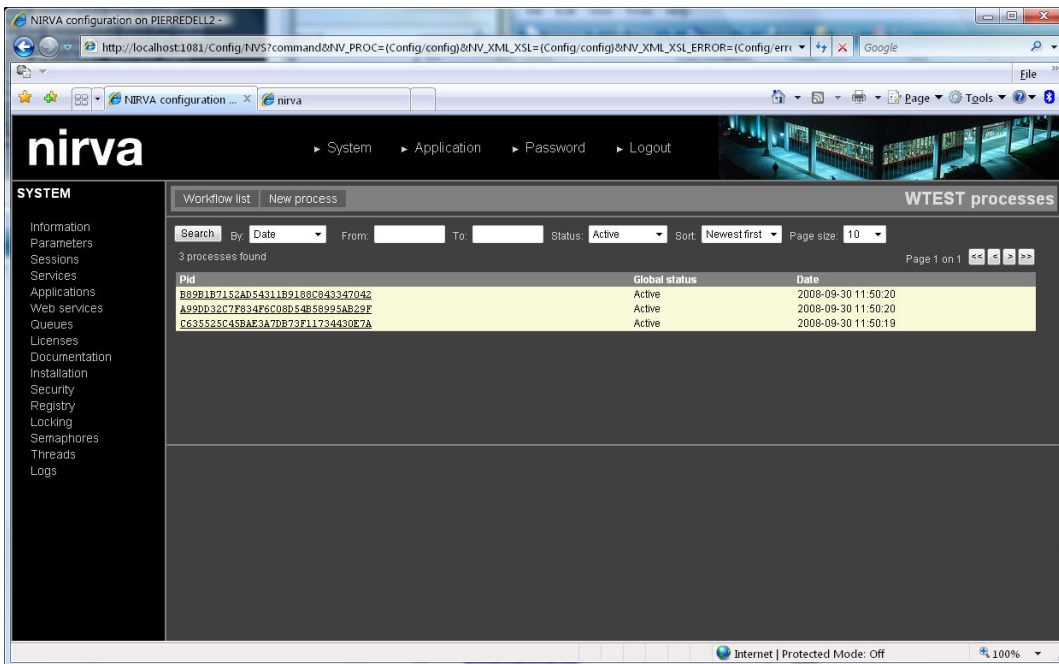
Start the application by pressing the  button on the NWTEST application line from the Applications list in System/Applications menu.

Using the application

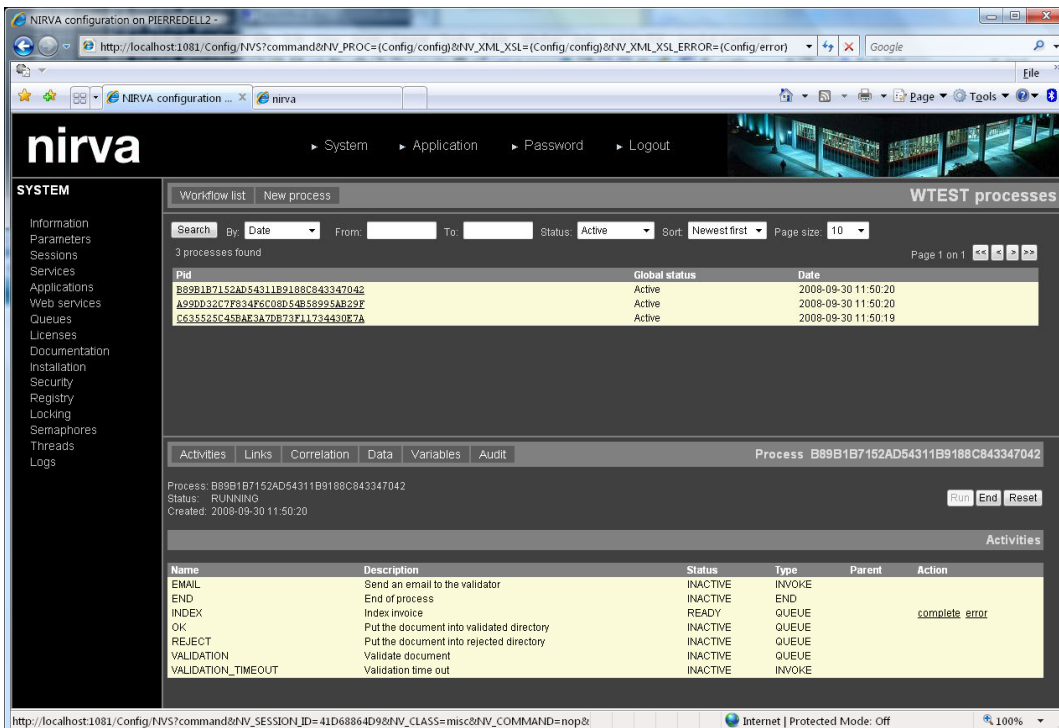
There are some example invoice files in the NWTEST/Images directory. Select Im3.pdf, Im6.pdf, Im12.pdf and copy them into the NWTEST/Input directory.

After few seconds, they should disappear from the input directory.

You can now have a look at the workflow service configuration. For this purpose go to Nirva configuration tool (login to the NWTEST application) and select the workflow service configuration. This displays the workflow list. Click on the workflow named "WTEST". This should display the following screen:



You can see three workflow processes that have been created. You can select one and display some details about it:

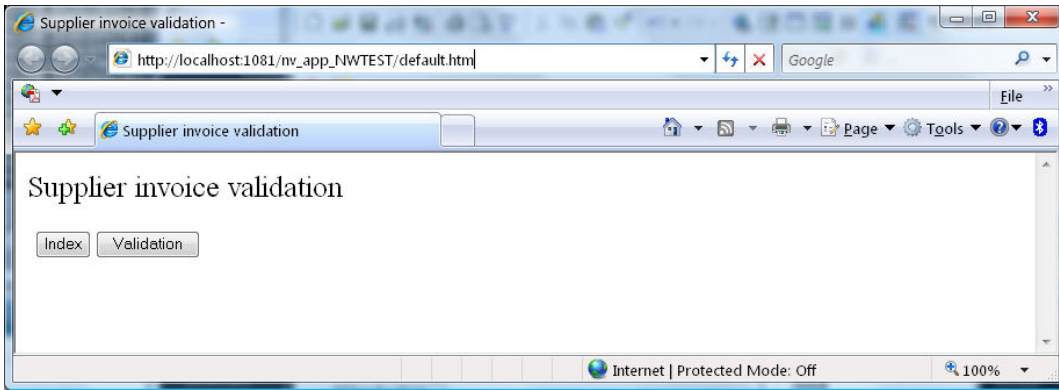


On the activities screen you can see that the process is ready for index now.

Please do not take any action for now in the bottom part of the screen.

In order to index the invoices, open the following url: http://localhost:1081/nv_app_NWTEST/default.htm

This displays the following screen:



Choose the index button in order to display the first invoice to index and to index them. Do the indexation in the following way:

Document	Supplier	Amount	Action	Reason
		20.47	Error	Unknown supplier
	BARCLAYS	693.99	Index	

Document	Supplier	Amount	Action	Reason
	bespoke-oak	5287.50	Index	

After the third document, a message indicates that there is no document left to index. Press the quit button to return to the selection screen.

Now if you look at the workflow processes from the workflow configuration, you'll see that one of the workflow processes is terminated so you have only two processes now displayed (press the search button if necessary to refresh the screen):

WTEST processes

2 processes found

Pid	Global status	Date
A99DD32C7F834F6C08D54B58995AB29F	Active	2008-09-30 11:50:20
C635525C45BAK3A7D873F11734430E7A	Active	2008-09-30 11:50:19

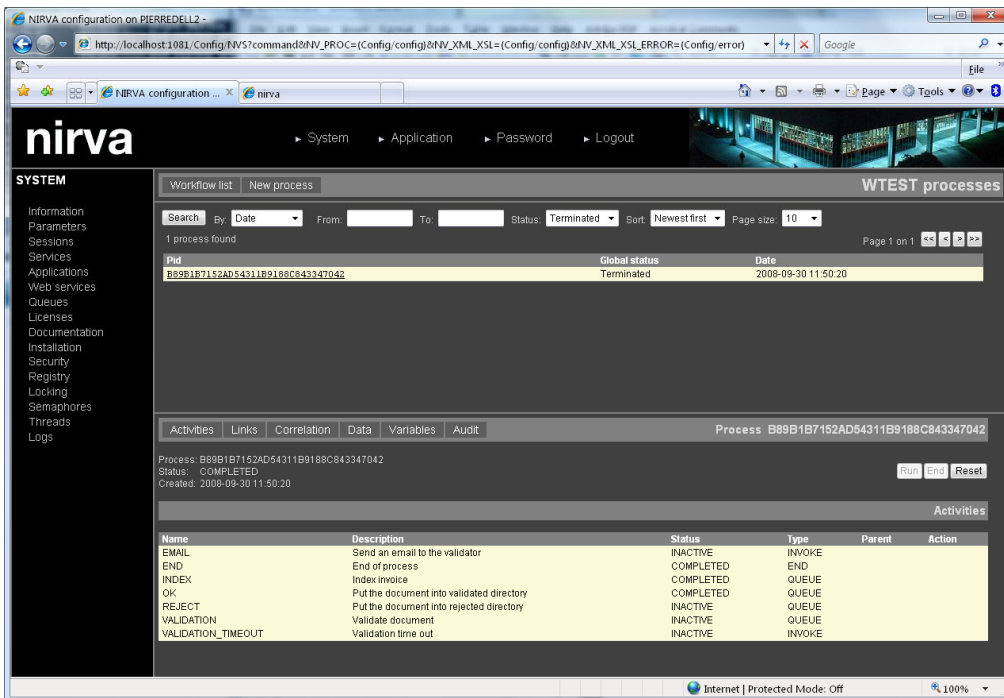
Process: A99DD32C7F834F6C08D54B58995AB29F

Status: RUNNING
Created: 2008-09-30 11:50:20

Name	Description	Status	Type	Parent	Action
EMAIL	Send an email to the validator	INACTIVE	INVOKE		
END	End of process	INACTIVE	END		
INDEX	Index invoice	COMPLETED	QUEUE		
OK	Put the document into validated directory	INACTIVE	QUEUE		
REJECT	Put the document into rejected directory	INACTIVE	QUEUE		
VALIDATION	Validate document	READY	QUEUE		complete error
VALIDATION_TIMEOUT	Validation time out	INACTIVE	INVOKE		

The terminated process is the Barclays one because the amount is less than 1000 and it has been indexed correctly. The workflow has sent it to the NWTEST/Output/Validated directory with an xml file containing the index. The terminated process is now a candidate for the purge mechanism and will be deleted after one day (retention time configured for this workflow).

You can see the Barclays process by selecting "Terminated" in the search criteria:



Now we are going to proceed to the validation step. For that come back to the application screen (http://localhost:1081/nv_app_NWTEST/default.htm) and select the validation button. Proceed to the validation in the following way:

Document	Supplier	Amount	Action	Reason
<p>A screenshot of a utility bill from 'Three'. The bill includes details such as 'Your bill period' (03/10/08 to 02/01/09), 'Your account number' (301753), and 'Your bill date' (22 Jan 09). It also lists 'Total charges before VAT' as £12.61 and 'Total charges after VAT' as £18.47. The bill is addressed to 'Mr. [redacted]' and includes a QR code.</p>	Three	20.47	Validate	
<p>A screenshot of an invoice from 'bespoke-oak.com'. The invoice is for 'Oak-Framed Extension' and requests 'final payment as agreed quotation'. The amount is listed as £4500.00. The invoice number is 63, dated 27/02/06.</p>	bespoke-oak	5287.50	Reject	Too expensive

After the second document, a screen displays that there is no document left to be validated. Just press the quit button to return to the selection screen.

Now if you come back to the workflow configuration, you can see that all workflows are terminated. If you go to the output directories, you'll see two documents in the Validated directory and one document in the Rejected directory.

Analysis

You need a text editor in this part to in order to look at the application code.

Input

The input module is made by a scheduled task named "input" that calls a perl procedure also named "input". It is called every 2 seconds.

This procedure polls the input directory for pdf files and for each file it creates a new workflow process. Creating the process is done by this code:

```
# Create a new workflow case
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CREATE| WORKFLOW=|WTEST| START=|NO|
NV_NO_ERROR=|YES|");

# Get the PID
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID| NV_NO_ERROR=|YES|");
my $PID = $NV::RESULT;

# Leave if not successful (we must have a PID if successful)
if($PID eq ""){return;}
```

The error management is reduced to the minimum here for simplification of the example. The process is created but not started immediately since we have some more work to do before.

Once the process has been created, we store the pdf file directly in the process data part in a Nirva file object named "INVOICE":

```
# Open the process data storage area and store the file into it as a file object named
INVOICE

# First rename the object
NV::Command("NV_CMD=|OBJECT:SET_NAME| NAME=|$OBJNAME| NEW_NAME=|INVOICE|");

# Open the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:OPEN_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");
```

```
# Write the file into it
NV::Command("NV_CMD=|REGISTRY:SET| SERVICE=|WORKFLOW| NAME=|WTEST:$PID|
ENTRIES=|INVOICE|");

# Close the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CLOSE_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Rename the object to the previous name
NV::Command("NV_CMD=|OBJECT:SET_NAME| NAME=|INVOICE| NEW_NAME=|$OBJNAME|");
```

The file is stored in the workflow process data here. It could be also stored into the Nirva storage service.

Now, we prepare some workflow process variables to store future index and also the original file name. We need the latter to correctly name the output file which will have the same name than the input file:

```
# Create workflow variables for the supplier, the amount, the reject reason and the
origin file name
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
VALUE=|0|");

NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
VALUE=||");

NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
VALUE=||");

NV::Command("NV_CMD=|OBJECT:FILE_GET_FILENAME| NAME=|$OBJNAME|");

NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|FILENAME|
VALUE=|#NV_RESULT|");
```

We then create a process variable containing the maximum amount authorized for automatically validation the invoice (set to 1000 here):

```
# Create workflow variables for the maximum amount to not validate
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|VAMOUNT|
VALUE=|1000|");
```

Finally we start the workflow process:

```
# Then start the workflow process
NV::Command("NV_CMD=|WORKFLOW:PROCESS:RUN| WORKFLOW=|WTEST| PID=|$PID|");
```

When the process starts, this automatically starts the index activity because we have defined this activity with "autostart" flag.

Index

The index activity is a Queue activity so that all documents to be indexed are placed in a queue and several indexers can call the workflow in order to get the next document to be indexed.

The workflow serializes the access to the queue insuring that a single process will be accessed by a single indexer.

The index module is done by a user interface and its associated perl procedure named “web_index”.

Code

When entering the module or when pressing the “Next doc” button, the code asks the workflow index activity queue for a new document:

```
# Asks the workflow for a process with an index activity to run
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:GET| WORKFLOW=|WTEST| ACTIVITY=|INDEX|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID|");
$PID = $NV::RESULT;

# Leave if no process available
if($PID eq "")
{
    # Set mode to NODOC
    NV::Command("NV_CMD=|VARIABLE:SET| NAME=|MODE| VALUE=|NODOC|");
    return;
}

# Open the process data storage area and retrieve the pdf stored as a file object named
INVOICE

# Open the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:OPEN_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Get the object
NV::Command("NV_CMD=|REGISTRY:GET| SERVICE=|WORKFLOW| NAME=|WTEST:$PID|
ENTRIES=|INVOICE| APPEND=|YES| REPLACE=|YES|");

# Close the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CLOSE_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Reset the amount and supplier variables
NV::Command("NV_CMD=|VARIABLE:REMOVE| NAME=|AMOUNT|");
NV::Command("NV_CMD=|VARIABLE:REMOVE| NAME=|SUPPLIER|");
```

In this code we get the process if there is one (it then goes from READY to RUNNING state) and we also get the stored pdf file in order to display the related document.

If the user indexes the document by pressing the index button, we store the index into the dedicated process variables and we signal the workflow that the activity is complete:

```
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
VALUE=|$AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
VALUE=|#SUPPLIER|");

# Then set the activity to complete state
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:COMPLETE| WORKFLOW=|WTEST| ACTIVITY=|INDEX|
PID=|$PID|");
```

In case of error we do almost the same but we add the reason to another process variable and we signal the workflow that the activity is in error:

```
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
VALUE=|$AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
VALUE=|$SUPPLIER|");

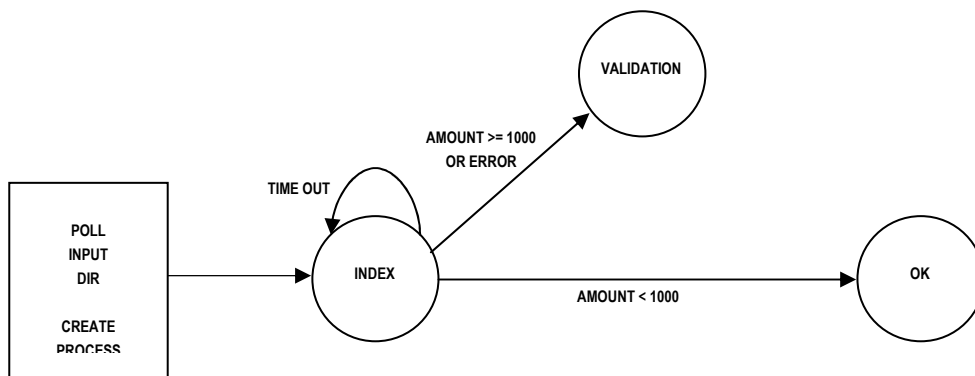
# Set also error reason
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
VALUE=|#REASON|");

# Then set the activity to error state
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:ERROR| WORKFLOW=|WTEST| ACTIVITY=|INDEX|
PID=|$PID|");
```

Finally if the user wants to interrupt the process and quit the application without indexing the document, we must resend the process (if there is one displayed) to the index queue:

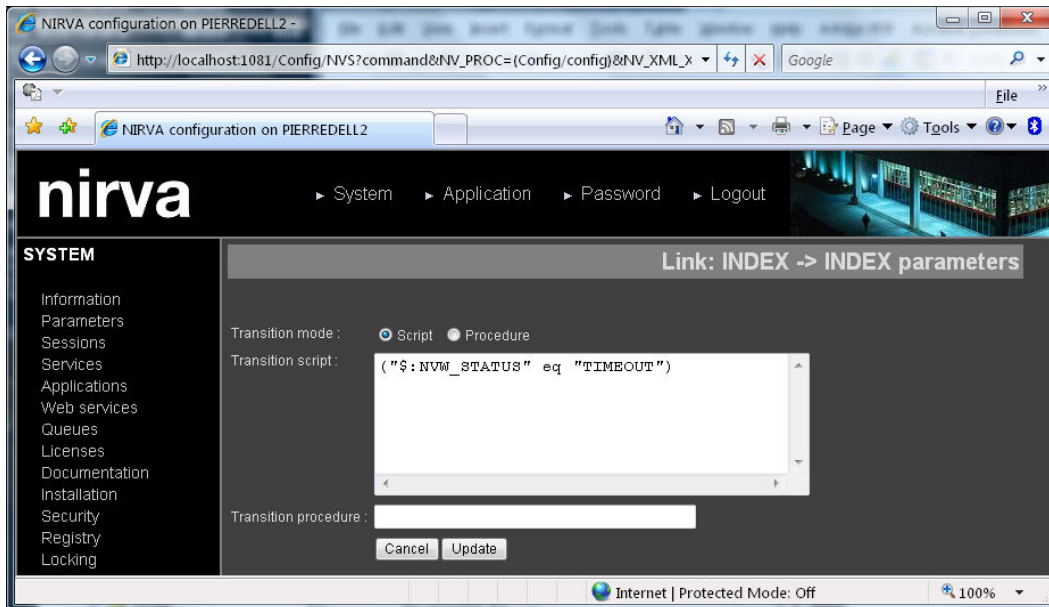
```
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:RESTART| WORKFLOW=|WTEST| ACTIVITY=|INDEX|
PID=|$PID|");
```

Workflow



We have three output links:

One is a self link that occurs in case of time out. This is to avoid a situation where the indexer is away from the application for a long time or if there is a failure on the workstation while indexing. The execution time out for the index activity has been set to 300 seconds. After this time, the time out management will send it back to the index queue. This is done automatically by defining the INDEX:INDEX link:



The transition script activates the link if the status of the activity is “TIMEOUT”.

Since the index activity join condition is empty, the activity is automatically activated (so placed in the queue) when one of the incoming links is active (the default join condition is a logical OR between incoming links).

The second link is INDEX:OK. It is simply defined with the following script:

```
("${NVW_STATUS}" eq "COMPLETED") && ($AMOUNT < $VAMOUNT)
```

So if the status is “COMPLETED” and the amount is less than 1000 (VAMOUNT has been defined to 1000 in the input module) then the document goes to the OK activity (validated). This allows validation of documents with a low amount. The maximum amount for automatic validation is hard coded but could easily be a parameter stored in the application registry.

The last link is INDEX:VALIDATION. It is defined with the following script:

```
("${NVW_STATUS}" eq "ERROR") || ($AMOUNT >= $VAMOUNT)
```

So if the document is in error or if the amount is greater than the maximum value it goes to the validation step.

Validation

The validation involves several activities: VALIDATION, VALIDATION_TIMEOUT and EMAIL.

The VALIDATION activity is a Queue activity so all the documents to be validated are placed in a queue and several supervisors can call the workflow in order to get the next document to be validated.

The workflow serializes the access to the queue assuming that a single process will be accessed by a single supervisor.

The validation module is done by a user interface and its associated perl procedure named "web_validation".

Code

When entering the module or when pressing the "Next doc" button, the code asks the workflow validation activity queue for a new document:

```
# Asks the workflow for a process with a validation activity to run
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:GET| WORKFLOW=|WTEST| ACTIVITY=|VALIDATION|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID|");
$PID = $NV::RESULT;

# Leave if no process available
if($PID eq "")
{
    # Set mode to NODOC
    NV::Command("NV_CMD=|VARIABLE:SET| NAME=|MODE| VALUE=|NODOC|");
    return;
}

# Open the process data storage area and retrieve the pdf stored as a file object named
INVOICE

# Open the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:OPEN_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Get the object
NV::Command("NV_CMD=|REGISTRY:GET| SERVICE=|WORKFLOW| NAME=|WTEST:$PID|
ENTRIES=|INVOICE| APPEND=|YES| REPLACE=|YES|");

# Close the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CLOSE_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Also get the amount, supplier and reason variables

NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
RESULT=|AMOUNT|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|AMOUNT| NV_VAR=|AMOUNT|");

NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
RESULT=|SUPPLIER|");
```

```
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|SUPPLIER| NV_VAR=|SUPPLIER|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
RESULT=|REASON|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|REASON| NV_VAR=|REASON|");
```

In this code we get the process if there is one (it then goes from READY to RUNNING state) and we also get the stored pdf file and the index values in order to display them.

If the user validates the document by pressing the validate button, we store the index into the dedicated process variables and we signal the workflow that the activity is complete:

```
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
VALUE=|$AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
VALUE=|#SUPPLIER|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
VALUE=||");

# Then set the activity to complete state
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:COMPLETE| WORKFLOW=|WTEST| ACTIVITY=|VALIDATION|
PID=|$PID|");
```

In case of rejection we do almost the same but we add the reason to another process variable and we signal the workflow that the activity is in error:

```
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
VALUE=|$AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
VALUE=|$SUPPLIER|");

# Set also error reason
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:SET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
VALUE=|#REASON|");

# Then set the activity to error state
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:ERROR| WORKFLOW=|WTEST| ACTIVITY=|VALIDATION|
PID=|$PID|");
```

Finally if the user wants to quit the application without indexing the document, we must resend the process (if there is one displayed) to the validation queue:

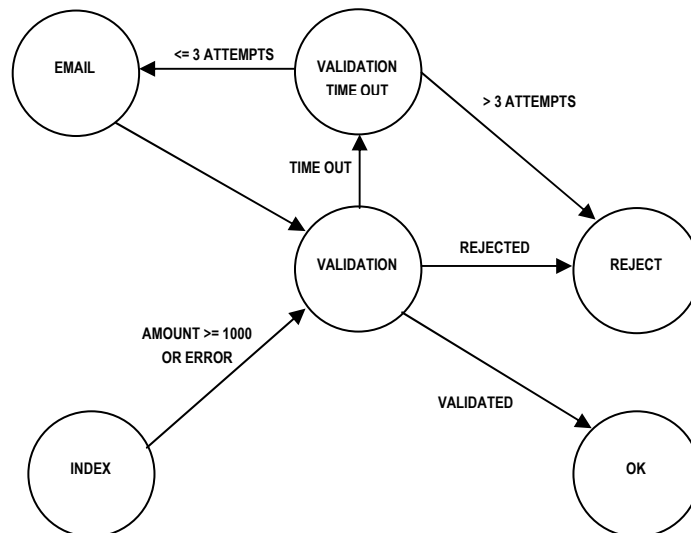
```
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:RESTART| WORKFLOW=|WTEST| ACTIVITY=|VALIDATION|
PID=|$PID|");
```

The VALIDATION_TIMEOUT code just displays information to the console. This is an Invoke activity. The code is in the "validation_timeout.pl" perl procedure.

The EMAIL code also just displays information to the console. It is possible to modify the code in order to send an actual email using the Nirva MAIL:SEND command.

This is an Invoke activity. The code is in the “email.pl” perl procedure.

Workflow



The VALIDATION activity has two input links and no join condition. So if any of the input link is active, the activity starts (the process is sent to the validation queue).

For this activity we have also defined two time outs. The first is a time out before the activity is retrieved from the queue (time out) and the second one is a time out when the activity has been retrieved but not completed (execution time out). The time out has been set to 60 minutes (3600 seconds) and the execution time out to 30 minutes (1800 seconds).

We have three output links for the VALIDATION activity:

The first output link is VALIDATION:OK. It is defined with the following script:

```
"$:NVW_STATUS" eq "COMPLETED"
```

So if the document is validated, the process goes to the OK activity.

The second output link is VALIDATION:REJECT. It is defined with the following script:

```
"$:NVW_STATUS" eq "ERROR"
```

So if the document is not validated, the process goes to the REJECT activity.

The last output link is VALIDATION:VALIDATION_TIMEOUT. It is defined with the following script:

```
$:NVW_STATUS eq "TIMEOUT"
```

So if the document is in time out, the process goes to the VALIDATION_TIMEOUT activity.

The VALIDATION_TIMEOUT activity has output links that verify the number of times the time out occurred. If it occurred more than three times, the document is rejected.

Here are the scripts of the VALIDATION_TIMEOUT output links:

```
VALIDATION_TIMEOUT:REJECT: "$:NVW_COUNT > 3"
```

```
VALIDATION_TIMEOUT:EMAIL: "$:NVW_COUNT <= 3"
```

NVW_COUNT is a standard activity variable giving the number of occurrence of an activity (here the current activity).

Ok

The OK activity is a Queue activity processed by a scheduled task. It just sends the invoice to the validated output directory, and creates the associated XML file containing the indexed information.

Code

The OK module code is inside the ok.pl perl procedure.

```
# Asks the workflow for a process with a ok activity to run
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:GET| WORKFLOW=|WTEST| ACTIVITY=|OK|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID|");
$PID = $NV::RESULT;

# Leave if no process available
if($PID eq ""){exit();}

# Open the process data storage area and retrieve the pdf stored as a file object named
INVOICE

# Open the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:OPEN_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Get the object
NV::Command("NV_CMD=|REGISTRY:GET| SERVICE=|WORKFLOW| NAME=|WTEST:$PID|
ENTRIES=|INVOICE|");

# Close the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CLOSE_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");
```

```

# Get the origin file name stored as a process variable during the input step
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID|
NAME=|FILENAME|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|RESULT|");
$FILENAME=$NV::RESULT;

# Copy the file to the output directory
NV::Command("NV_CMD=|OBJECT:CREATE| TYPE=|FILE| NAME=|DEST| PERSIST=|-1|
FILENAME=|$OUTPUT_DIR/$FILENAME| REPLACE=|YES|");
NV::Command("NV_CMD=|OBJECT:COPY| SNAME=|INVOICE| NAME=|DEST|");

# Also create an xml file that contains the document index
NV::Command("NV_CMD=|OBJECT:FILE_GET_FILENAME| NAME=|DEST| WITH_EXT=|NO|");
$XMLFILENAME="$OUTPUT_DIR/$NV::RESULT.xml";
NV::Command("NV_CMD=|OBJECT:CREATE| TYPE=|FILE| NAME=|OUTPUT| PERSIST=|-1|
FILENAME=|$XMLFILENAME| REPLACE=|YES|");

# Get the amount stored as a process variable during the index step
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
RESULT=|AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
RESULT=|SUPPLIER|");

# Populate the xml file
NV::Command("NV_CMD=|XML:SET_XML| OUTPUT=|BINARY| OBJECTS=|AMOUNT;SUPPLIER|");
NV::Command("NV_CMD=|XML:TRANSFORM| XSL_NAME=|output| XMLDEST=|OUTPUT|");

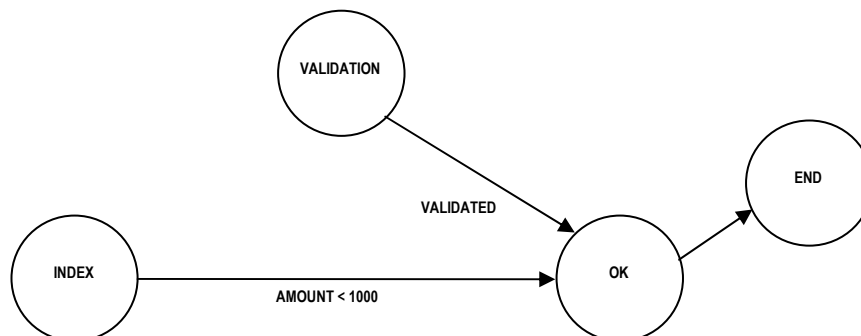
# Set the activity to success
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:COMPLETE| WORKFLOW=|WTEST| ACTIVITY=|OK|
PID=|$PID|");

# Stop if there is a global stop request
NV::Command("NV_CMD=|SESSION:CHECK_CLOSE_REQUEST|");
if($NV::RESULT eq "YES")
{
    exit();
}

```

In this code we loop asking the workflow if there are some documents in the queue and we process them. The last part of the loop is necessary in Nirva when we have a loop that may be long. We must then check if there is no external request to stop the session. Otherwise, Nirva may hang while stopping.

Workflow



The OK activity join condition has been defined in the following way:

```
:INDEX || :VALIDATION
```

So it will start if one of the INDEX or VALIDATION input links are activated. In fact it is not necessary to create a join condition here because the default is a logical OR between incoming links. This has been done only to show an example of join condition.

The only output link is OK:END that terminates the process.

Reject

The REJECT activity is a Queue activity processed by a scheduled task. It just sends the invoice to the rejected output directory, and creates the associated XML file containing the index and the reject reason. If the reject comes from a time out, we set the reject reason to “Time out in validation”.

Code

The REJECT module code is inside the reject.pl perl procedure.

```

# Asks the workflow for a process with a reject activity to run
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:GET| WORKFLOW=|WTEST| ACTIVITY=|REJECT|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|PID|");
$PID = $NV::RESULT;

# Leave if no process available
if($PID eq ""){exit();}

# Open the process data storage area and retrieve the pdf stored as a file object named
INVOICE

# Open the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:OPEN_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");
  
```

```

# Get the object
NV::Command("NV_CMD=|REGISTRY:GET| SERVICE=|WORKFLOW| NAME=|WTEST:$PID|
ENTRIES=|INVOICE|");

# Close the workflow process registry
NV::Command("NV_CMD=|WORKFLOW:PROCESS:CLOSE_REGISTRY| WORKFLOW=|WTEST| PID=|$PID|");

# Get the origin file name stored as a process variable during the input step
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID|
NAME=|FILENAME|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|RESULT|");
$FILENAME=$NV::RESULT;

# Copy the file to the output directory
NV::Command("NV_CMD=|OBJECT:CREATE| TYPE=|FILE| NAME=|DEST| PERSIST=|-1|
FILENAME=|$OUTPUT_DIR/$FILENAME| REPLACE=|YES|");
NV::Command("NV_CMD=|OBJECT:COPY| SNAME=|INVOICE| NAME=|DEST|");

# Also create an xml file that contains the document index and reject reason
NV::Command("NV_CMD=|OBJECT:FILE_GET_FILENAME| NAME=|DEST| WITH_EXT=|NO|");
$xmlFILENAME="$OUTPUT_DIR/$NV::RESULT.xml";
NV::Command("NV_CMD=|OBJECT:CREATE| TYPE=|FILE| NAME=|OUTPUT| PERSIST=|-1|
FILENAME=|$xmlFILENAME| REPLACE=|YES|");

# Get the index information stored as process variables
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|AMOUNT|
RESULT=|AMOUNT|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|SUPPLIER|
RESULT=|SUPPLIER|");
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID| NAME=|REASON|
RESULT=|REASON|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|REASON|");
$REASON=$NV::RESULT;

# If the validation step is in time out, we must set the reason to time out
NV::Command("NV_CMD=|WORKFLOW:VARIABLE:GET| WORKFLOW=|WTEST| PID=|$PID|
ACTIVITY=|VALIDATION| NAME=|NVW_STATUS|");
NV::Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|RESULT|");
if($NV::RESULT eq "TIMEOUT")
{
    $REASON = "Time out in validation";
}

# If the reason is not know, we set an unknown reason
if($REASON eq "")
{
    $REASON = "Unknown reason";
}
NV::Command("NV_CMD=|OBJECT:STRING_SET_VALUE| NAME=|REASON| VALUE=|$REASON|");

# Populate the xml file
NV::Command("NV_CMD=|XML:SET_XML| OUTPUT=|BINARY| OBJECTS=|AMOUNT;SUPPLIER;REASON|");
NV::Command("NV_CMD=|XML:TRANSFORM| XSL_NAME=|output| XMLDEST=|OUTPUT|");

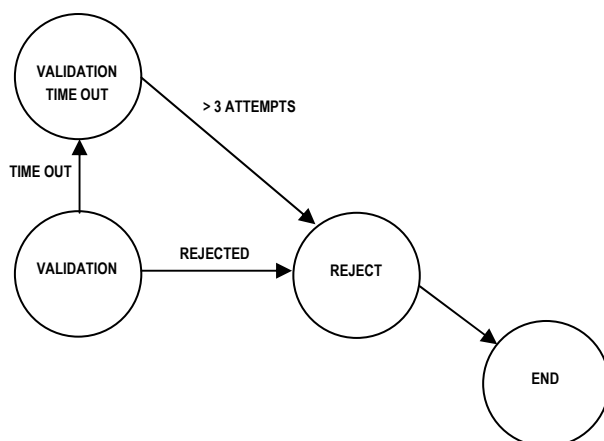
```

```
# Set the activity to success
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:COMPLETE| WORKFLOW=|WTEST| ACTIVITY=|REJECT|
PID=|$PID|");

# Stop if there is a global stop request
NV::Command("NV_CMD=|SESSION:CHECK_CLOSE_REQUEST|");
if($NV::RESULT eq "YES")
{
    exit();
}
```

In this code we loop asking the workflow if there are some documents in the queue and if yes, we process them. The last part of the loop is necessary in Nirva when we have a loop that may be long. We must then check if there is no external request to stop the session. Otherwise, Nirva may hang while stopping.

Workflow



The REJECT activity has no join condition so the default is a logical OR between all incoming links.

The only output link is REJECT:END that terminates the process.

Other modules

There is some other small part of code around the application.

Init and cleanup code

This code is not detailed here because it is not relevant to the workflow. The init code just installs necessary components if they are not yet installed (init.pl procedure).

Triggers

Some activity triggers have been defined at workflow level. These triggers just display a message in the console when an activity starts or ends or when a process is purged.

Listeners

As for all workflows, some minimum listeners or scheduled tasks must be defined for procession time out, recovery and purge mechanisms.

The code of these listeners is very simple:

Recovery:

```
# Just call the PROCESS:RECOVERY command for this workflow
NV::Command("NV_CMD=|WORKFLOW:PROCESS:RECOVERY| WORKFLOW=|WTEST|");
```

Time out:

```
# Just call the ACTIVITY:TIME_OUT command for this workflow
NV::Command("NV_CMD=|WORKFLOW:ACTIVITY:TIME_OUT| WORKFLOW=|WTEST|");
```

Purge:

```
# Just call the PROCESS:PURGE command for this workflow
NV::Command("NV_CMD=|WORKFLOW:PROCESS:PURGE| WORKFLOW=|WTEST|");
```

The retention time of the workflow has been set to 1 day (workflow level parameter).

Reference

This chapter gives the complete reference of all the WORKFLOW service commands.

Classes

Here are the available WORKFLOW service classes:

Class	Description
WORKFLOW	Workflow level commands
PROCESS	Process level commands
ACTIVITY	Activity level commands
LINK	Link level commands
VARIABLE	Commands for manipulating process and activity variables
PRODUCTION	Commands for accessing production lists by date
CORRELATION	Correlation tables commands

Error codes

WORKFLOW Class

Value	Description
101	Invalid command
102	Invalid parameter
103	Invalid workflow name
104	The workflow does not exist
105	Another user has requested to stop the workflow
106	Cannot start the workflow

Value	Description
107	Cannot stop the workflow (probably in use)
108	The workflow ever exists
109	Cannot create the workflow
110	The workflow must be stopped first
111	Cannot remove the workflow
112	Unknown workflow parameter
113	Cannot access workflow data directory
114	Cannot get workflow (the workflow does not exist or is disabled)
115	Operation authorized only from outside the workflow
116	Not enough memory
117	The workflow has no end activity
118	The block activity has no end activity
119	The workflow has no activity to start

PROCESS Class

Value	Description
101	The process does not exist or has been removed
102	Cannot update the process status
103	Cannot set the variable
104	Cannot remove the variable
105	Cannot create the process
106	Cannot start the process
107	Cannot reset the process
108	The process is not running
109	Cannot get audit data

ACTIVITY Class

Value	Description
101	Invalid activity name
102	The activity ever exists
103	Cannot create activity
104	Activity does not exist
105	Cannot load activity queue from disk

Value	Description
106	Cannot update activity status
107	Cannot remove the activity from the queue
108	Cannot add the activity to the queue
109	The operation is not supported for this activity
110	The activity is not in a running state
111	The activity is not in a ready state
112	No queue found for this activity
113	Cannot update activity reason
114	The activity queue is empty
115	Error when processing an activity time out
116	The parent activity does not exist
117	The parent activity is not a block activity
118	The block activity has no activity to start
119	The end or error activity has outgoing links
120	Cannot update activity time out

LINK Class

Value	Description
101	Invalid link source
102	Invalid link target
103	The link ever exists
104	Cannot create link
105	The link does not exist
106	Cannot activate the link
107	Cannot inactivate the link
108	The source or target activity does not exist
109	The link is outside a block activity

CORRELATION Class

Value	Description
101	Cannot set the correlation data
102	Cannot remove the correlation data
103	Cannot create the correlation table
104	Cannot remove the correlation table

Value	Description
105	Cannot get the correlation table
106	Cannot change the correlation table parameters

Permissions

Name	Description
ADMIN	Administration. Allows workflow creation, start stop and change parameters.
WORK	Allows working with workflows. This permission is necessary for workflow users. Without it the workflow process state cannot be changed.

Commands

For each command, the reference gives the command name, the sources for which the command may be used, the command description, the eventual command permissions, the parameter list and the eventual list of objects created by the command.



The parameters described in this chapter are command specific parameters. For general parameters, please refer to the Nirva command syntax chapter.

The available sources are:

- Client for all Nirva client interfaces including Nirva client library (nvc).
- Web for commands from a web browser.
- Procedure for commands from a Nirva procedure.
- Service for commands from service to service

WORKFLOW class

The WORKFLOW class provides commands at workflow level.

CREATE

WORKFLOW:WORKFLOW:CREATE

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	No
Service		

Description

Creates a new workflow. A workflow is identified by its name that must be unique.

Permissions

ADMIN

Parameters

WORKFLOW	Workflow name. Uniquely identifies the workflow. Cannot contain spaces or special characters. Mandatory parameter.
DESCRIPTION	Workflow description.
DIR	Base directory for the workflow data. If the given directory does not exist, the command creates it.
ACTIVITY_START_TRIGGER_PROC	Name of a trigger procedure that will run each time an activity is started. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
ACTIVITY_STOP_TRIGGER_PROC	Name of a trigger procedure that will run each time an activity end. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
PROCESS_PURGE_TRIGGER_PROC	Name of a trigger procedure that will run each time a process is removed from the system. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
RETENTION_DELAY	Retention delay in days for the processes of the workflow. After the retention delay, a process is removed from the system by the purge mechanism. If this value is 0, the process is removed immediately when it ends. This is the default value.

Objects created

None

DISABLE

WORKFLOW:WORKFLOW:DISABLE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Disables a workflow. No process can be created if the workflow is disabled.

Permissions

ADMIN

Parameters

WORKFLOW Workflow name.

Objects created

None

ENABLE

WORKFLOW:WORKFLOW:ENABLE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Enables a workflow. No process can be created if the workflow is disabled.

This command verifies the workflow (see the VERIFY command) and returns an error if the verification is not successful.

Permissions

ADMIN

Parameters*WORKFLOW* Workflow name.**Objects created**

None

EXPORT

WORKFLOW:WORKFLOW:EXPORT

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Saves the workflow definition parameters to a file allowing further installation on another computer using the IMPORT command. The saved format is XML.

Permissions

None

Parameters*WORKFLOW* Workflow name.*RESULT* Name of the file result object. The default is WORKFLOW_FILE.**Objects created**

WORKFLOW_FILE Workflow export file in the requested format. This is a Nirva file object. The object is created by the command if it does not exist. The name of the object can be changed using the RESULT parameter.

GET_PARAM

WORKFLOW:WORKFLOW:GET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves a workflow parameter. This command creates a string object in the output container. The name of the parameter to retrieve is given in the PARAM parameter. This can be one of the following values:

- *DESCRIPTION* Workflow description.
- *DIR* Base directory for the workflow data.
- *ACTIVITY_START_TRIGGER_PROC* Activity start trigger procedure.
- *ACTIVITY_STOP_TRIGGER_PROC* Activity stop trigger procedure.
- *PROCESS_PURGE_TRIGGER_PROC* Process purge trigger procedure.
- *RETENTION_DELAY* Process retention delay in days.

Permissions

None

Parameters

- WORKFLOW* Workflow name.
- PARAM* Parameter name.
- RESULT* Name of the string result object. The default is *WORKFLOW_PARAM*.

Objects created

- WORKFLOW_PARAM* Value of the requested parameter. This is a Nirva string object. The name of this object can be changed using the *RESULT* parameter.

IMPORT

WORKFLOW:WORKFLOW:IMPORT

Source	Use Input Container	Use Output Container
Client Web Procedure Service	Yes	No

Description

Imports the workflow definition parameters from a previously exported file. The target workflow must be disabled otherwise this command fails. If the workflow does not exist, it is created.

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	The workflow name is inside the XML data but may be changed resulting in a copy of the workflow under a different name. If this parameter is not provided, the service uses the name from the import XML file.
<i>WORKFLOW_FILE</i>	Workflow import file in XML format resulting of a WORKFLOW:EXPORT command. This is a Nirva file object. The default value is "WORKFLOW_FILE".
<i>DIR</i>	Base directory for the workflow data. This parameter is only used if the imported workflow does not exist. If the given directory does not exist, the command creates it.

Objects created

None

LIST

WORKFLOW:WORKFLOW:LIST

Source	Use Input Container	Use Output Container
--------	---------------------	----------------------

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Lists all or a single workflow. The list is sent to a Nirva table object in the output container. This table contains the following columns:

- *NAME* Workflow name.
- *DESCRIPTION* Workflow description.
- *ENABLE* Flag telling if the workflow is enabled (YES) or not (NO).
- *BASEDIR* Base directory for the workflow data.
- *ACTIVITY_START_TRIGGER_PROC* Activity start trigger procedure.
- *ACTIVITY_STOP_TRIGGER_PROC* Activity stop trigger procedure.
- *PROCESS_PURGE_TRIGGER_PROC* Process purge trigger procedure.
- *RETENTION_DELAY* Process retention delay in seconds.
- *NUMUSERS* Number of users currently using this workflow.

Permissions

None

Parameters

- WORKFLOW* Workflow name. If this parameter is not given, the command gets the entire workflow list.
- RESULT* Name of the table result object. The default is *WORKFLOW_LIST*.

Objects created

- WORKFLOW_LIST* Workflow list. This is a Nirva table object. The name of this object can be changed using the *RESULT* parameter.

REMOVE

WORKFLOW:WORKFLOW:REMOVE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Removes an existing workflow. This completely removes the workflow and all its associated data from the system. If the workflow is currently running (enabled), the command fails.

Permissions

ADMIN

Parameters

WORKFLOW Workflow name.

Objects created

None

SET_PARAM

WORKFLOW:WORKFLOW:SET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Sets a workflow parameter. The name of the parameter to set is given in the PARAM parameter. This can be one of the following values:

- *DESCRIPTION* Workflow description.
- *DIR* Base directory for the workflow data.
- *ACTIVITY_START_TRIGGER_PROC* Activity start trigger procedure.
- *ACTIVITY_STOP_TRIGGER_PROC* Activity stop trigger procedure.
- *PROCESS_PURGE_TRIGGER_PROC* Process purge trigger procedure.
- *RETENTION_DELAY* Process retention delay in days.

See the WORKFLOW:CREATE command for further information on workflow parameters.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

- WORKFLOW* Workflow name.
- PARAM* Parameter name.
- VALUE* Parameter value.

Objects created

None

VERIFY

WORKFLOW:WORKFLOW:SET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Verifies the workflow activities and links and generate an error if something is bad (for example if a workflow has no end activity or if a link goes from outside to inside a bloc activity, etc.).

Permissions*None***Parameters***WORKFLOW* Workflow name.**Objects created***None***PROCESS class**

The PROCESS class provides commands at process level.

AUDIT

WORKFLOW:PROCESS:AUDIT

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Gets all or part of the audit data corresponding to the process.

Permissions*None***Parameters***WORKFLOW* Workflow name.*PID* Process ID.

PAGE Page number to get (default is 1). If page number is 0, nothing is returned (useful if only the information part is requested).

PAGE_SIZE Page size (default is 20). If the page size is 0, all the audit data is retrieved.

- AUDIT_DATA* Name of the audit data returned object. The default is "AUDIT_DATA".
- INFO* Allows giving information about the total number of audit lines. Can be YES or NO. The default is NO.
- AUDIT_INFO* Name of the resulting object containing audit information when INFO has been set to YES. Default is AUDIT_INFO.

Objects created

- AUDIT_DATA* This is a Nirva table object containing the audit data. The name of the object can be changed using the AUDIT_DATA parameter. It contains 4 columns: DATE (date/time on the form YYYY-MM-DD HH:MM:SS), ACTIVITY (activity name), ACTION and INFO (additional information).
- AUDIT_INFO* This is a Nirva indexed string list object containing information about the audit data. The command returns it only if the INFO parameter has been set to YES. The name of the object can be changed using the AUDIT_INFO parameter. It has the following keys: TOTAL is the total number of audit lines, PAGES is the number of pages, PAGE_SIZE is the page size.

CLOSE_REGISTRY

WORKFLOW:PROCESS:CLOSE_REGISTRY

Source	Use Input Container	Use Output Container
Client	No	No
Web		
Procedure		
Service		

Description

Closes the registry associated to the process previously opened by the OPEN_REGISTRY command. The registry name is composed of the workflow name followed by a : character and then the process Id. Once opened, it can be accessed like any user registry using Nirva registry commands. This is considered as a service registry and the service name must be set to "WORKFLOW" when accessing this registry.

There must be the same number of CLOSE_REGISTRY than OPEN_REGISTRY. If the session terminates, all opened registries owned by the session are automatically closed.

Permissions

None

Parameters

WORKFLOW Workflow name.
PID Process ID.

Objects created

None

CREATE

WORKFLOW:PROCESS:CREATE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Creates a new process for a given workflow. A new PID is created and returned as a string object in the output container.

Permissions

WORK

Parameters

WORKFLOW Workflow name.
PID Name of the resulting object. The default is PID.
START Start option. If this parameter is set to "YES", the process is immediately started. Otherwise, in order to start the process, a PROCESS:RUN command must be sent.

Objects created

PID PID of the created workflow. This is a Nirva string object.

END

WORKFLOW:PROCESS:END

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Explicitly stops a process. The process is then passed to the purge mechanism. The process status is set to STOPPED. The normal way to stop a process is to define a STOP or END activity. Using the PROCESS:END command should be reserved for maintenance.

Permissions

WORK

Parameters

WORKFLOW Workflow name.
PID Process ID.

Objects created

None

INFO

WORKFLOW:PROCESS:INFO

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Gets part or all information related to a process. This includes the activity and link lists with status. Following the requested information, the INFO command may return the following objects:

ACTIVITIES – Activity list for the process. This is a table object with the following columns:

■ <i>NAME</i>	Activity name.
■ <i>DESCRIPTION</i>	Activity description.
■ <i>TYPE</i>	Activity type. Can be “INVOKE”, “QUEUE”, “RECEIVE”, “WAIT”, “END”, “ERROR” or “BLOCK”.
■ <i>STATUS</i>	Activity status. Can be “INACTIVE”, “READY”, “RUNNING”, “COMPLETED”, “ERROR” or “INTERRUPTED”.
■ <i>PROC</i>	Activity procedure (invoke type only).
■ <i>REASON</i>	Error reason when status is ERROR.
■ <i>PARENT</i>	Parent activity.
■ <i>START_TRIGGER_PROC</i>	Start trigger procedure.
■ <i>STOP_TRIGGER_PROC</i>	Stop trigger procedure.
■ <i>EVAL_MODE</i>	Evaluation mode for establishing list of next activities to start and inactivate. This can be “LINK_TABLE” or “PROCEDURE”.
■ <i>EVAL_PROC</i>	Evaluation procedure when evaluation mode is “PROCEDURE”.
■ <i>JOIN_MODE</i>	Join condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.
■ <i>JOIN_PROC</i>	Join condition procedure when join mode is “PROCEDURE”.
■ <i>JOIN_SCRIPT</i>	Join script when join mode is “SCRIPT”.
■ <i>AUTOSTART</i>	Autostart flag. Can be “YES” or “NO”.

LINKS – Link list for the process. This is a table object with the following columns:

■ <i>SOURCE</i>	Source activity.
■ <i>TARGET</i>	Target activity.
■ <i>ACTIVE</i>	Active flag. Set to “YES” when the link is active and to “NO” otherwise.
■ <i>TRANSITION_MODE</i>	Transition condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.
■ <i>TRANSITION_PROC</i>	Transition condition procedure when transition mode is “PROCEDURE”.
■ <i>TRANSITION_SCRIPT</i>	Transition script when transition mode is “SCRIPT”.

INFO – Process information. This is an indexed string list object with the following keys:

- *PID* Process ID.
- *WORKFLOW* Workflow name.
- *STATUS* Process status. Can be INACTIVE, RUNNING, ERROR, COMPLETED or STOPPED.
- *CREATED* Creation date/time on format yyyy-mm-dd hh:mm:ss.
- *TERMINATED* Termination date/time on format yyyy-mm-dd hh:mm:ss. Empty if the process is not terminated.

Permissions

None

Parameters

- WORKFLOW* Workflow name.
- PID* Process ID.
- WHAT* Information to get. This is a combination of values separated by a semicolon (“;”) character. The possible values are “ACTIVITIES” for activity list, “LINKS” for link list, “INFO” for information about process or “ALL” for all the information (default).

Objects created

- ACTIVITIES* Process activity list. Table object. (See description of the command).
- LINKS* Process link list. Table object. (See description of the command).
- INFO* Process information. Indexed string list object. (See description of the command).

OPEN_REGISTRY

WORKFLOW:PROCESS:OPEN_REGISTRY

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Opens the registry associated to the process. The registry name is composed of the workflow name in uppercase followed by a : character and then the process Id. Once opened, it can be accessed like any user registry using Nirva registry commands. This is considered as a service registry and the service name must be set to "WORKFLOW" when accessing this registry.

After finishing working with a process registry, this one must be closed using the CLOSE_REGISTRY command. There must have the same number of CLOSE_REGISTRY than OPEN_REGISTRY. If the session terminates, all opened registries owned by the session are automatically closed.

Permissions

WORK

Parameters

WORKFLOW Workflow name.

PID Process ID.

Objects created

None

PURGE

WORKFLOW:PROCESS:PURGE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Retrieves and removes the processes that ended (COMPLETED, ERROR or STOPPED state) and for which the retention delay has expired.

This command must be called by the purge listener or scheduled task. It is useful only when a retention delay has been defined for the workflow.

The command is implemented as a loop that retrieves any processes to purge and purges them. The input and output containers are cleared after each process is purged.

Permissions

WORK

Parameters*WORKFLOW* Workflow name.**Objects created***None***RECOVERY**

WORKFLOW:PROCESS:RECOVERY

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Recovers all processes that were running when an accidental power off or crash occurs. This command must be called from the recovery scheduled task.

The command is implemented as a loop that retrieves any processes eligible for recovery and recovers them. The input and output containers are cleared after each process is recovered.

Permissions

WORK

Parameters*WORKFLOW* Workflow name.**Objects created***None*

RESET

WORKFLOW:PROCESS:RESET

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Resets a process to the state it was after creation and optionally restart it.

The following actions are made:

- All activities are turned to INACTIVE state
- All queue activities are removed from corresponding queues
- All links are set to INACTIVE
- All variables are reset to the values at the time of the first process start
- The process status is changed to INACTIVE or RUNNING if start option is set

The audit information and correlation data is kept.

The process data is also kept.

If the start option of the command is set, the process restarts otherwise a START command must be sent.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process ID.
<i>START</i>	Start option. If this parameter is set to "YES", the process is immediately restarted. Otherwise, in order to restart the process, a PROCESS:RUN command must be sent.

Objects created

None

RUN

WORKFLOW:PROCESS:RUN

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Starts a process previously created with the CREATE command. If the process is already running, the command does nothing and does not return any error. Otherwise, if the process is not in an inactive state, the command fails.

Permissions

WORK

Parameters

WORKFLOW Workflow name.
PID Process ID to run.

Objects created*None***ACTIVITY class**

The ACTIVITY class provides commands at activity level.

COMPLETE

WORKFLOW:ACTIVITY:COMPLETE

Source	Use Input Container	Use Output Container
--------	---------------------	----------------------

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Sets status of COMPLETED for an activity of type QUEUE or RECEIVE. The workflow then turns the status of the activity to COMPLETED, evaluates activities to start next and starts them.

Permissions

WORK

Parameters

WORKFLOW Workflow name.

PID Process ID.

ACTIVITY Activity name.

Objects created

None

CREATE

WORKFLOW:ACTIVITY:CREATE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Creates a new activity. An activity is identified by its name that must be unique.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>ACTIVITY</i>	Activity name.
<i>DESCRIPTION</i>	Activity description.
<i>TYPE</i>	Activity type. This must be "INVOKE", "QUEUE", "RECEIVE", "WAIT", "END", "ERROR" or "BLOCK". The default is "INVOKE".
<i>PROC</i>	Procedure for an invoke activity. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
<i>PARENT</i>	Name of the parent activity for an activity inside a block activity. This parameter allows defining all the activities inside a block.
<i>START_TRIGGER_PROC</i>	Name of a trigger procedure that will run each time the activity is started. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
<i>STOP_TRIGGER_PROC</i>	Name of a trigger procedure that will run each time the activity ends. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
<i>EVAL_MODE</i>	Evaluation mode for the next activities to start. This can be "LINK_TABLE" for using the link table or "PROCEDURE" for using an external procedure.
<i>EVAL_PROC</i>	Evaluation procedure when the EVAL_MODE has been set to "PROCEDURE". The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide.
<i>JOIN_MODE</i>	Join mode. Can be set to "SCRIPT" for using a script or to "PROCEDURE" for using an external procedure.
<i>JOIN_PROC</i>	Join procedure when the JOIN_MODE has been set to "PROCEDURE". The content of this parameter is similar to the NV_PROC parameter described in the Nirva user's guide. The join procedure receives three parameters named NVW_WORKFLOW, NVW_PID and NVW_ACTIVITY corresponding to workflow name, process ID and activity name. It must set a session variable named "NVW_RESULT" that can be set to the values "TRUE" or "FALSE" following the result of the join condition.
<i>JOIN_SCRIPT</i>	Join script when the JOIN_MODE has been set to "SCRIPT". The join script is a test that is automatically evaluated by the workflow. It must be written in perl. The activity and process variables can be used but generally the join script will use only link variables. Link variables are boolean variables having the format :SOURCE_ACTIVITY where SOURCE_ACTIVITY is the name of the source activity. For example ":A1 && :A2" is a valid script that will

validate the join condition only if the links from activity A1 and A2 are validated.

AUTOSTART Flag indicating whether the activity must be started when starting the process or the block. If this parameter is set to “YES” and the activity has a parent (part of a block activity), it will be started automatically when the parent activity starts. If this parameter is set to “YES” and the activity has no parent, it will be started automatically when the process starts

TIME_OUT Time out value in seconds for a Wait or Receive activity to remain in RUNNING state and for a queue activity to remain in READY state. When this value is 0, no time out occurs.

EXEC_TIME_OUT Time out value in seconds for a queue activity to remain in RUNNING state. When this value is 0, no time out occurs.

Objects created

None

ERROR

WORKFLOW:ACTIVITY:ERROR

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Allows the reporting of an error status for an activity of type QUEUE or RECEIVE. The workflow then sets the status of the activity to ERROR, evaluates the activities to start next and starts them.

Permissions

WORK

Parameters

WORKFLOW Workflow name.

PID Process ID.

ACTIVITY Activity name.

REASON Error reason.

Objects created

None

GET

WORKFLOW:ACTIVITY:GET

Source	Use Input Container	Use Output Container
Client	No	Yes
Web		
Procedure		
Service		

Description

Retrieves a QUEUE activity from the queue. This command can be typically used by a listener that processes a queue activity. The listener issue an ACTIVITY:GET command to retrieve the PID of an activity of QUEUE type, processes it and then issue ACTIVITY:COMPLETED or ACTIVITY:ERROR when finished.

Permissions

WORK

Parameters

WORKFLOW Workflow name.

ACTIVITY Activity name. This must be queue activity

PID Process ID. If this parameter is given, the workflow will get the corresponding PID.

ERROR_IF_EMPTY Flag indicating whether the command must generate an error when the queue is empty. Can be "YES" or "NO". The default is "NO".

RESULT Name of the string result object. The default is PID.

Objects created

PID Process ID. Empty if the command failed or didn't find any process ID. This is a Nirva string object.

GET_PARAM

WORKFLOW:ACTIVITY:GET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves an activity parameter. This command creates a string object in the output container. The name of the parameter to retrieve is given in the PARAM parameter. This can be one of the following values:

- *DESCRIPTION* Activity description.
- *PROC* Activity procedure (invoke type only).
- *TYPE* Activity type. Can be "INVOKE", "QUEUE", "RECEIVE", "WAIT", "END", "ERROR" or "BLOCK".
- *STATUS* Activity status. Can be "INACTIVE", "READY", "RUNNING", "COMPLETED" or "ERROR".
- *PARENT* Parent activity.
- *START_TRIGGER_PROC* Start trigger procedure.
- *STOP_TRIGGER_PROC* Stop trigger procedure.
- *EVAL_MODE* Evaluation mode for establishing list of next activities to start and inactivate. This can be "LINK_TABLE" or "PROCEDURE".
- *EVAL_PROC* Evaluation procedure when evaluation mode is "PROCEDURE".
- *JOIN_MODE* Join condition evaluation mode. This can be "SCRIPT" or "PROCEDURE".
- *JOIN_PROC* Join condition procedure when join mode is "PROCEDURE".
- *JOIN_SCRIPT* Join script when join mode is "SCRIPT".
- *AUTOSTART* Autostart flag. Can be "YES" or "NO".
- *TIME_OUT* Time out value in seconds for a wait or receive activity to remain in RUNNING state and for a queue activity to remain in READY state.
- *EXEC_TIME_OUT* Time out value in seconds for a queue activity to remain in RUNNING state.

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>ACTIVITY</i>	Workflow name.
<i>PARAM</i>	Parameter name.
<i>RESULT</i>	Name of the string result object. The default is <i>ACTIVITY_PARAM</i> .

Objects created

<i>ACTIVITY_PARAM</i>	Value of the requested parameter. This is a Nirva string object. The name of this object can be changed using the <i>RESULT</i> parameter.
-----------------------	--

LIST**WORKFLOW:ACTIVITY:LIST**

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Lists all or a single activity. The list is sent to a Nirva table object in the output container. This table contains the following columns:

- *NAME* Activity name.
- *DESCRIPTION* Activity description.
- *TYPE* Activity type. Can be "INVOKE", "QUEUE", "RECEIVE", "WAIT", "END", "ERROR" or "BLOCK".
- *STATUS* Activity status. Can be "INACTIVE", "READY", "RUNNING", "COMPLETED" or "ERROR".
- *PROC* Activity procedure (invoke type only).
- *REASON* Error reason when status is ERROR.
- *PARENT* Parent activity.
- *START_TRIGGER_PROC* Start trigger procedure.
- *STOP_TRIGGER_PROC* Stop trigger procedure.

- *EVAL_MODE* Evaluation mode for establishing list of next activities to start and inactivate. This can be "LINK_TABLE" or "PROCEDURE".
- *EVAL_PROC* Evaluation procedure when evaluation mode is "PROCEDURE".
- *JOIN_MODE* Join condition evaluation mode. This can be "SCRIPT" or "PROCEDURE".
- *JOIN_PROC* Join condition procedure when join mode is "PROCEDURE".
- *JOIN_SCRIPT* Join script when join mode is "SCRIPT".
- *AUTOSTART* Autostart flag. Can be "YES" or "NO".
- *TIME_OUT* Time out value in seconds for a wait or receive activity to remain in RUNNING state and for a queue activity to remain in READY state.
- *EXEC_TIME_OUT* Time out value in seconds for a queue activity to remain in RUNNING state.

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process ID. If this parameter is not given, the command gets the activity from the workflow activity list (used in configuration). When a valid PID is given, the command returns the activity table of the PID with all its actual status.
<i>ACTIVITY</i>	Activity name. If this parameter is not given, the command gets the entire activity list.
<i>RESULT</i>	Name of the table result object. The default is <i>ACTIVITY_LIST</i> .

Objects created

<i>ACTIVITY_LIST</i>	Activity list. This is a Nirva table object. The name of this object can be changed using the <i>RESULT</i> parameter.
----------------------	--

REMOVE

WORKFLOW:ACTIVITY:REMOVE

Source	Use Input Container	Use Output Container
--------	---------------------	----------------------

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Remove an existing activity.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

WORKFLOW Workflow name.

ACTIVITY Activity name.

Objects created

None

RESTART

WORKFLOW:ACTIVITY:RESTART

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Resends a Queue activity back to the queue after an ACTIVITY:GET command. The activity is placed at the front of the queue.

If the activity is not a Queue activity, the command fails.

If the activity is not in READY or RUNNING state, the command does nothing.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process ID.
<i>ACTIVITY</i>	Activity name.

Objects created

None

SET_PARAM

WORKFLOW:ACTIVITY:SET_PARAM

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	No
Service		

Description

Set an activity parameter. The name of the parameter to set is given in the PARAM parameter. This can be one of the following values:

- *DESCRIPTION* Activity description.
- *PROC* Activity procedure (invoke type only).
- *TYPE* Activity type. Can be "INVOKE", "QUEUE", "RECEIVE", "WAIT", "END", "ERROR" or "BLOCK".
- *PARENT* Parent activity.
- *START_TRIGGER_PROC* Start trigger procedure.
- *STOP_TRIGGER_PROC* Stop trigger procedure.
- *EVAL_MODE* Evaluation mode for establishing list of next activities to start and inactivate. This can be "LINK_TABLE" or "PROCEDURE".
- *EVAL_PROC* Evaluation procedure when evaluation mode is "PROCEDURE".

- *JOIN_MODE* Join condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.
- *JOIN_PROC* Join condition procedure when join mode is “PROCEDURE”.
- *JOIN_SCRIPT* Join script when join mode is “SCRIPT”.
- *AUTOSTART* Autostart flag. Can be “YES” or “NO”.
- *TIME_OUT* Time out value in seconds for a wait or receive activity to remain in RUNNING state and for a queue activity to remain in READY state.
- *EXEC_TIME_OUT* Time out value in seconds for a queue activity to remain in RUNNING state.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

- WORKFLOW* Workflow name.
- ACTIVITY* Activity name.
- PARAM* Parameter name.
- VALUE* Parameter value.

Objects created

None

SET_TIME_OUT

WORKFLOW:ACTIVITY:SET_TIME_OUT

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Dynamically change the activity time out for an existing process. The new time out value will be taken in care only the next time the activity goes to the READY or RUNNING state.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process ID.
<i>ACTIVITY</i>	Activity name.
<i>TIME_OUT</i>	Time out value in seconds for a wait or receive activity to remain in RUNNING state and for a queue activity to remain in READY state. If this parameter is not provided, the time out is not changed.
<i>EXEC_TIME_OUT</i>	Time out value in seconds for a queue activity to remain in RUNNING state. If this parameter is not provided, the exec time out is not changed. If the activity is not a QUEUE activity, the exec time out is not set.

Objects created

None

TIME_OUT

WORKFLOW:ACTIVITY:TIME_OUT

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	No
Service		

Description

This command must be called by the time out listener of the workflow. It checks all the process activities that are in time out, change their status to TIMEOUT (Queue and Receive activities) or COMPLETED (Wait activity) evaluates the next activities to start and starts them.

The command is implemented as a loop that gets any processes in time out and releases it. The input and output containers are cleared after each process released.

Permissions

WORK

Parameters

WORKFLOW Workflow name.

Objects created

None

LINK class

The LINK class provides commands at link level.

CREATE

WORKFLOW:LINK:CREATE

Source	Use Input Container	Use Output Container
Client	No	No
Web		
Procedure		
Service		

Description

Creates a new link. A link is identified by its activity source and target names. The pair must be unique.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

WORKFLOW Workflow name.

SOURCE Source activity name.

- TARGET** Target activity name.
- TRANSITION_MODE** Transition mode. Can be set to “SCRIPT” for using a script or to “PROCEDURE” for using an external procedure.
- TRANSITION_PROC** Transition procedure when the TRANSITION_MODE has been set to “PROCEDURE”. The content of this parameter is similar to the NV_PROC parameter described in the Nirva user’s guide. The join procedure receives four parameters named NVW_WORKFLOW, NVW_PID, NVW_SOURCE, NVW_TARGET corresponding to workflow name, process ID, activity source and target names. It must set a session variable named “NVW_RESULT” that can be set to the values “TRUE” or “FALSE” following the result of the transition condition.
- TRANSITION_SCRIPT** Transition script when the TRANSITION_MODE has been set to “SCRIPT”. The transition script is a test that is automatically evaluated by the workflow. It must be written in perl. The activity and process variables can be used. An activity variable has the format \$ACTIVITY:VARIABLE where ACTIVITY is the activity name and VARIABLE is the variable name (ex.: \$A1:VAR1). If the activity name is not given (but the : character is given), this refers to the source activity (ex.: \$:VAR1). A process variable has the format \$VARNAME (ex.: \$VAR1).

Objects created

None

GET_PARAM

WORKFLOW:LINK:GET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves a link parameter. This command creates a string object in the output container. The name of the parameter to retrieve is given in the PARAM parameter. This can be one of the following values:

- **TRANSITION_MODE** Transition condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.

- *TRANSITION_PROC* Transition condition procedure when transition mode is “PROCEDURE”.
- *TRANSITION_SCRIPT* Transition script when transition mode is “SCRIPT”.

Permissions

None

Parameters

- WORKFLOW* Workflow name.
- SOURCE* Source activity name.
- TARGET* Target activity name.
- PARAM* Parameter name.
- RESULT* Name of the string result object. The default is LINK_PARAM.

Objects created

- LINK_PARAM* Value of the requested parameter. This is a Nirva string object. The name of this object can be changed using the RESULT parameter.

LIST

WORKFLOW:LINK:LIST

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Lists all or a single link. The list is sent to a Nirva table object in the output container. This table contains the following columns:

- *SOURCE* Source activity.
- *TARGET* Target activity.
- *ACTIVE* Active flag. Set to “YES” when the link is active and to “NO” otherwise.

- *TRANSITION_MODE* Transition condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.
- *TRANSITION_PROC* Transition condition procedure when transition mode is “PROCEDURE”.
- *TRANSITION_SCRIPT* Transition script when transition mode is “SCRIPT”.

Permissions

None

Parameters

- WORKFLOW* Workflow name.
- PID* Process ID. If this parameter is not given, the command gets the link from the workflow link list (used in configuration). When a valid PID is given, the command returns the link table of the PID with all its actual status.
- SOURCE* Source activity name. If this parameter is not given, the command gets the entire link list.
- TARGET* Target activity name. If this parameter is not given, the command gets the entire link list.
- RESULT* Name of the table result object. The default is LINK_LIST.

Objects created

- LINK_LIST* Link list. This is a Nirva table object. The name of this object can be changed using the RESULT parameter.

REMOVE

WORKFLOW:LINK:REMOVE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Removes an existing link.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

WORKFLOW Workflow name.
SOURCE Source activity name.
TARGET Target activity name.

Objects created

None

SET_PARAM

WORKFLOW:LINK:SET_PARAM

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Sets a link parameter. The name of the parameter to set is given in the PARAM parameter. This can be one of the following values:

- *TRANSITION_MODE* Transition condition evaluation mode. This can be “SCRIPT” or “PROCEDURE”.
- *TRANSITION_PROC* Transition condition procedure when transition mode is “PROCEDURE”.
- *TRANSITION_SCRIPT* Transition script when transition mode is “SCRIPT”.

The command fails if the workflow is currently in use (enabled).

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>SOURCE</i>	Source activity name.
<i>TARGET</i>	Target activity name.
<i>PARAM</i>	Parameter name.
<i>VALUE</i>	Parameter value.

Objects created

None

VARIABLE class

The VARIABLE class provides commands to set or get process or activity variables.

GET

WORKFLOW:VARIABLE:GET

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves a process or activity variable. The result is returned into a string list object.

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process Id.
<i>ACTIVITY</i>	Activity name. Must be left blank to access a process variable.
<i>NAME</i>	Variable name.

RESULT Name of the string result object. The default is RESULT.

Objects created

RESULT Requested variable value. This is a Nirva string object. The name of this object can be changed using the RESULT parameter.

LIST

WORKFLOW:VARIABLE:LIST

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Lists process and activity variables. The result is returned into a table object having three columns: NAME, VALUE and ACTIVITY. For process level variables, the ACTIVITY field is blank.

Permissions

None

Parameters

WORKFLOW Workflow name.

PID Process Id.

RESULT Name of the table result object. The default is VARIABLES.

Objects created

VARIABLES Variable names and values. This is a Nirva table object. The name of this object can be changed using the RESULT parameter.

SET

WORKFLOW:VARIABLE:SET

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	No
Service		

Description

Sets a process or activity variable.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process Id.
<i>ACTIVITY</i>	Activity name. Must be left blank to set a process variable.
<i>NAME</i>	Variable name.
<i>VALUE</i>	Variable value.

Objects created

None

REMOVE

WORKFLOW:VARIABLE:REMOVE

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	Yes
Service		

Description

Removes a process or activity variable.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process Id.
<i>ACTIVITY</i>	Activity name. Must be left blank to remove a process variable.
<i>NAME</i>	Variable name.

Objects created

None

PRODUCTION class

The PRODUCTION class provides commands to access the list of process.

SEARCH

WORKFLOW:PRODUCTION:SEARCH

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	Yes
Service		

Description

Returns a list of process given date/time based search criteria. This command should not be used to often because it can slow the system (requires database synchronization).

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>FROM</i>	<p>From date/time criteria. The general format of the DATE parameter is YYYY-MM-DD HH:MM:SS where DD is the day from 1 to 31, MM is the month from 1 to 12, YYYY is the complete year, HH the hour, MM the minutes and SS the seconds. The time part is not mandatory.</p> <p>The FROM parameter accepts another format that is a relative date before the actual date. The format is then the following: $-x_y$ where x is an integer and y a letter that can take the value d for days, h for hours, m for minutes and s for seconds. For example “-2h” will be the actual time minus 2 hours.</p>
<i>TO</i>	<p>To date/time criteria. The general format of the DATE parameter is YYYY-MM-DD HH:MM:SS where DD is the day from 1 to 31, MM is the month from 1 to 12, YYYY is the complete year, HH the hour, MM the minutes and SS the seconds. The time part is not mandatory.</p> <p>The TO parameter accepts another format that is a relative date before the actual date. The format is then the following: $-x_y$ where x is an integer and y a letter that can take the value d for days, h for hours, m for minutes and s for seconds. For example “-2h” will be the actual time minus 2 hours.</p>
<i>WHAT</i>	<p>This can restrict the search to active or terminated processes. The WHAT parameter can take the values ACTIVE for active processes (default), TERMINATED for all terminated processes, COMPLETED for processes terminated successfully, ERROR for processes terminated with error, STOPPED for stopped processes or ALL for all kinds of processes.</p> <p>Compatibility issue: for process created with a workflow service version prior to 3.06, the COMPLETED, ERROR and STOPPED values will return empty results.</p>
<i>PID</i>	This can restrict the search to the specific process ID.
<i>SORT</i>	The result is always sorted by date. The SORT parameter gives the sort order. This can be ASC for ascending (default) or DESC for descending.
<i>PAGE</i>	Page number to get (default is 1). If page number is 0, nothing is returned (useful if only the information part is requested).
<i>PAGE_SIZE</i>	Page size (default is 20). If the page size is 0, all the records are retrieved (this can be long if many processes).
<i>PROCESS_LIST</i>	Name of the resulting object containing found processes. Default is PROCESS_LIST.
<i>INFO</i>	Allows giving information about the total number of documents corresponding to the search criteria. Can be YES or NO. The default is NO.
<i>QUERY_INFO</i>	Name of the resulting object containing query information when INFO has been set to YES. Default is QUERY_INFO.

Objects created

PROCESS_LIST

This is a Nirva table object containing the found processes. The name of the object can be changed using the PROCESS_LIST parameter. It contains four columns PID (Process ID), DATE (Creation date on the form YYYY-MM-DD HH:MM:SS), EDATE (Termination date on the form YYYY-MM-DD HH:MM:SS or NULL if the process is active) and STATUS (global status 0 for active, 1 for terminated with success, 2 for terminated with error and 3 for stopped).

Compatibility issue: for process created with a workflow service version prior to 3.06, the processes terminated with error or the stopped processes will return 1 as global status.

QUERY_INFO

This is a Nirva indexedstringlist object containing some information about the query. The SEARCH command returns it only if the INFO parameter has been set to YES. The name of the object can be changed using the QUERY_INFO parameter. It has the following keys: TOTAL is the total number of found processes, PAGES is the number of pages, PAGE_SIZE is the page size, FROM is the FROM date criteria, TO is the TO date criteria, WORKFLOW is the workflow name, WHAT is the WHAT criteria.

CORRELATION class

The CORRELATION class provides correlation commands. Correlation is business data that can be associated to process IDs in order to retrieve them from business information.

CREATE_ENTRY

WORKFLOW:CORRELATION:CREATE_ENTRY

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Associate a process to a correlation value.

Permissions

WORK

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process Id.
<i>TABLE</i>	Correlation table name.
<i>VALUE</i>	Correlation value.

Objects created

None

CREATE_TABLE

WORKFLOW:CORRELATION:CREATE_TABLE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	No

Description

Creates a new correlation table. This command works only if the workflow is not running (disabled).

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>TABLE</i>	Correlation table name. The correlation table name should not contain any space or special character. The correlation table name is not case sensitive.
<i>DESCRIPTION</i>	Correlation table description.

Objects created

None

GET_PIDS

WORKFLOW:CORRELATION:GET_PIDS

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves the process IDs associated to a given correlation value.

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>TABLE</i>	Correlation table name.
<i>VALUE</i>	Correlation value.
<i>RESULT</i>	Name of the table result object. The default is PIDS.

Objects created

<i>PIDS</i>	Found PIDs. This is a Nirva string list object. The name of this object can be changed using the <i>RESULT</i> parameter.
-------------	---

GET_VALUES

WORKFLOW:CORRELATION:GET_VALUES

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Retrieves all the correlation data associated to a given process. The result is sent to a Nirva table object in the output container. This table contains the following columns:

- *TABLE* Correlation table name.
- *VALUE* Correlation value.

Permissions

None

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>PID</i>	Process Id.
<i>RESULT</i>	Name of the table result object. The default is <i>CORRELATION_VALUES</i> .

Objects created

CORRELATION_VALUES Correlation data associated to the process. This is a Nirva table object. The name of this object can be changed using the *RESULT* parameter.

LIST_TABLES

WORKFLOW:CORRELATION:LIST_TABLES

Source	Use Input Container	Use Output Container
Client Web Procedure Service	No	Yes

Description

Lists all or a single correlation table. The list is sent to a Nirva table object in the output container. This table contains the following columns:

- *TABLE* Correlation table name.
- *DESCRIPTION* Correlation table description.

Permissions*None***Parameters***WORKFLOW* Workflow name.*RESULT* Name of the result object. The default is CORRELATION_TABLE_LIST.**Objects created***CORRELATION_TABLE_LIST* Table list. This is a Nirva table object. The name of this object can be changed using the RESULT parameter.

REMOVE_ENTRY

WORKFLOW:CORRELATION:REMOVE_ENTRY

Source	Use Input Container	Use Output Container
Client		
Web		
Procedure	No	No
Service		

Description

Removes the association between a process and a correlation value.

Permissions

WORK

Parameters*WORKFLOW* Workflow name.*PID* Process Id.*TABLE* Correlation table name.*VALUE* Correlation value.**Objects created***None*

REMOVE_TABLE

WORKFLOW:CORRELATION:REMOVE_TABLE

Source	Use Input Container	Use Output Container
Client	No	No
Web		
Procedure		
Service		

Description

Removes a correlation table. This command works only if the workflow is not running (disabled).

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>TABLE</i>	Correlation table name. The correlation table should not contain any space or special character. The correlation table name is not case sensitive.

Objects created

None

SET_TABLE_PARAM

WORKFLOW:CORRELATION:SET_TABLE_PARAM

Source	Use Input Container	Use Output Container
Client	No	No
Web		
Procedure		
Service		

Description

Changes the correlation table parameters. This command works only if the workflow is not running (disabled).

Permissions

ADMIN

Parameters

<i>WORKFLOW</i>	Workflow name.
<i>TABLE</i>	Correlation table name. The correlation table should not contain any space or special character. The correlation table name is not case sensitive.
<i>DESCRIPTION</i>	Correlation table description.

Objects created

None