

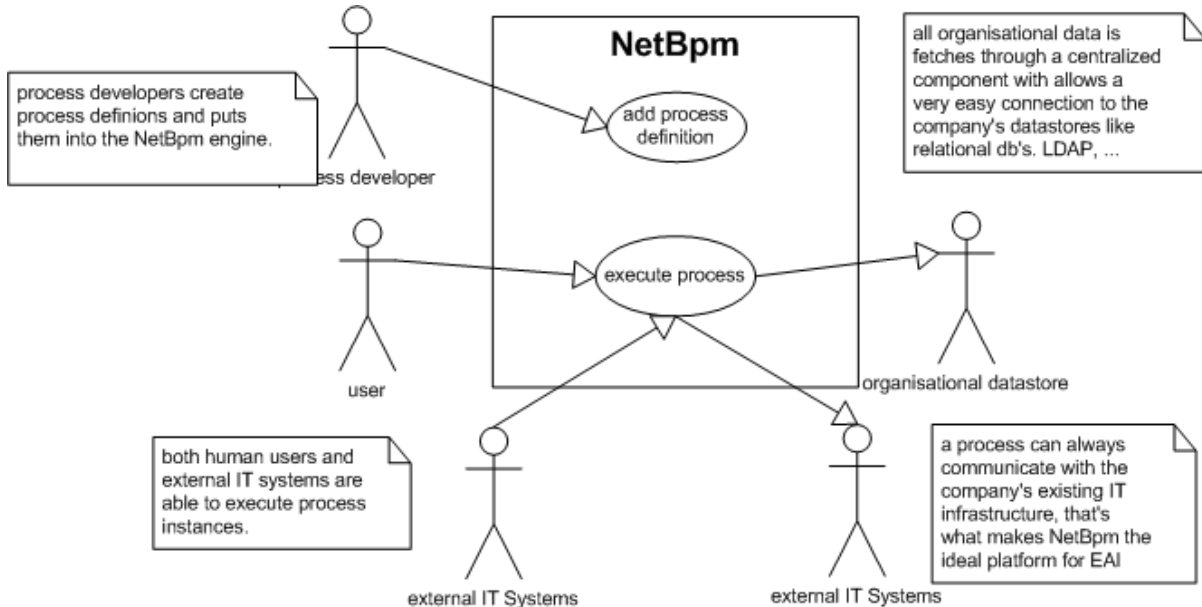
The NetBpm Architecture

Table of contents

1 The component interfaces.....	2
1.1 The definition component.....	4
1.2 The execution component.....	4
1.3 The organisation component.....	5
1.4 The log component.....	6
1.5 The scheduler component.....	6
1.6 The admin component.....	6
2 State.....	6
3 Attributes.....	7
3.1 Serialization.....	7
3.2 Scope.....	8
4 The delegation principle.....	8
4.1 The delegation interfaces.....	9
5 ProcessInvocationHandler.....	10
6 NetBpm as integration platform.....	10
6.1 Process initiated interactio.....	10
6.2 Integration initiated by an external application.....	11
7 Exception handling.....	11

1. The component interfaces

NetBpm is developed as a set of components. Each component is exposed with a central interface (session facade). The functionality of the different components is inspired by the WfMC.



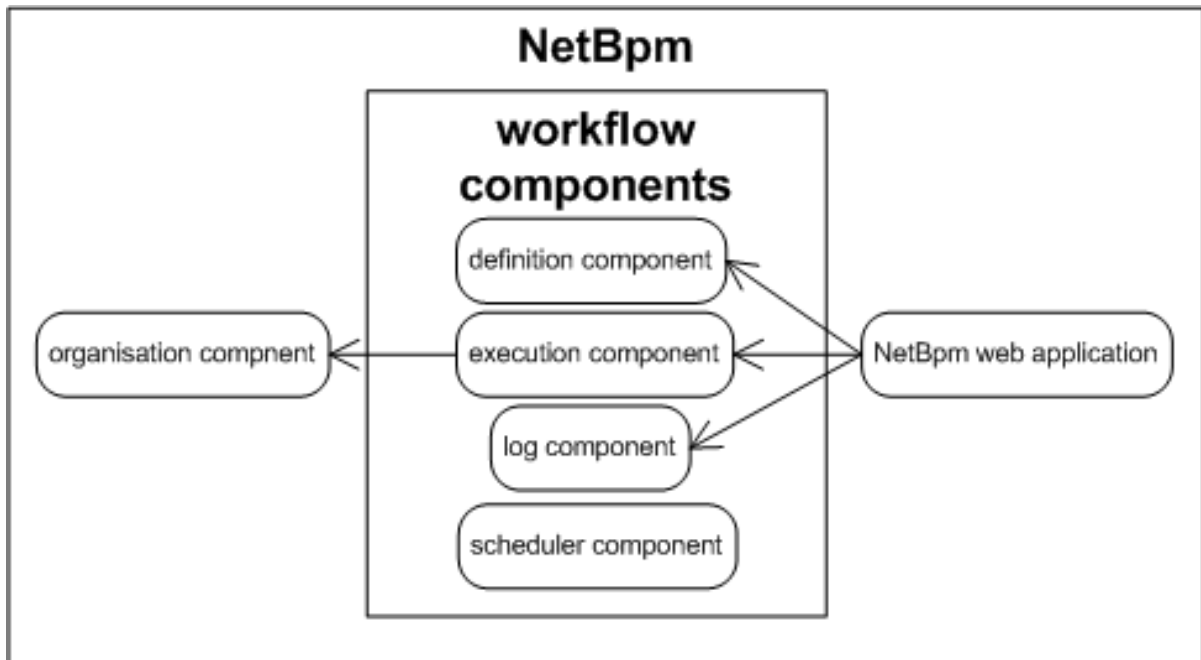
The NetBpm interfaces

The NetBpm interfaces

1. The interface for the **'process developer'** : This interface is used to put process definitions into the NetBpm engine. First, process developers create a process definition as explained in [nPdI](#). The result is a process archive. A process archive is a complete specification of a business process. The interface described here allows to put a process archive into the NetBpm engine through e.g. the NetBpm web interface for developers. The process archive is parsed and stored in the NetBpm database.
2. The interface for a **'user'** : The term 'user' denotes people that execute processes. The 2 main actions of the process execution are start a process and perform an activity. Starting a process creates a new process instance. A process instance is one execution of a process definition. A process instance is composed of one or more flows-of-execution that execute in parallel. For each flow that is in an activity-state, exactly one person or unit is responsible for the performing the activity. Performing the activity is the second action that can be done with the execution interface. The execution interface also allows additional actions such as getting a tasklist, getting the list of available process

definitions, ...

3. The interface between '**external IT systems**' and the NetBpm engine : External IT-systems can interact with NetBpm in 2 ways :
 - System initiated interaction : When a system wants to initiate an action upon a process, it has to use the execution interface as described above.
 - Process initiated interaction : For interactions that need to be initiated by the process, **&Interactors&** are used. Interactors are part of the process definition and included in the process archive. Interactors are .NET-classes that can access the FlowContext. That enables Interactors to establish a two-way communication between the process-specific information (FlowContext) and the external IT-systems.
4. The interface between NetBpm and the '**organisational datastore**' : For several calculations, NetBpm needs information about the organisation : people, teams, departments, roles, ... To allow NetBpm to be quickly implemented in an organisation, all this information is collected though one central organisation component. For different organisations, this kind of information is stored in different types of datastores like e.g. an LDAP-system, a relational user database, an exchange server, ... The use of a so called session facade pattern, makes it easy to connect all data sources of organisational information to NetBpm.



The NetBpm component structure

The NetBpm component structure

1.1. The definition component

1.2. The execution component

The 2 most typical execution actions are starting a process instance and performing an activity.

1.2.1. Starting a process instance

These are the methods for starting a process instance, declared in `NetBpm.Workflow.Execution.ExecutionComponent`

```

public interface ExecutionComponent {
    ...
    IProcessInstance startProcessInstance(Int64
processDefinitionId);

    IProcessInstance startProcessInstance(Int64
processDefinitionId,
                                         IDictionary attributeValues);

    IProcessInstance startProcessInstance(Int64
processDefinitionId,
                                         IDictionary attributeValues,
                                         String transitionName);

    IProcessInstance startProcessInstance(Int64
processDefinitionId,
                                         IDictionary attributeValues,
                                         String transitionName,
                                         Relations relations);
    ...
}

```

Methods for starting a process instance

Note that the implementation of the 4 methods is the same and that the first three methods only provide default values for the parameters.

processDefinitionId identifies the process definition for which an instance should be created.

attributeValues is a Map of attribute-names to attribute-values. The values provided in this Map will be stored in the attribute instance variables. Before the attributesValues are stored in the attribute instances, every attribute instance is initialized to the initial value (if one is specified).

transitionName is the name of the transition to take. In case there is only one leaving transition on the start-activity, it is selected automatically and does not have to be specified.

relations specifies which related objects of the returned process-instance should be resolved. This prohibits that for each request, all objects that can be reached from the process instance are fetched from the database, optionally serialized and send to the client. e.g. If you want to access the attribute values, proceed as follows :

FIXME (pbolle):

add examplecode

Example of using Relations

1.2.2. Starting a process instance

```
public interface ExecutionComponent {
...
    IList performActivity(Int64 flowId);
    IList performActivity(Int64 flowId,
        IDictionary attributeValues);
    IList performActivity(Int64 flowId,
        IDictionary attributeValues,
        String transitionName);
    IList performActivity(Int64 flowId,
        IDictionary attributeValues,
        String transitionName,
        Relations relations);
...
}
```

Methods for starting a process instance

Note that the implementation of the 4 methods is the same and that the first three methods only provide default values for the parameters.

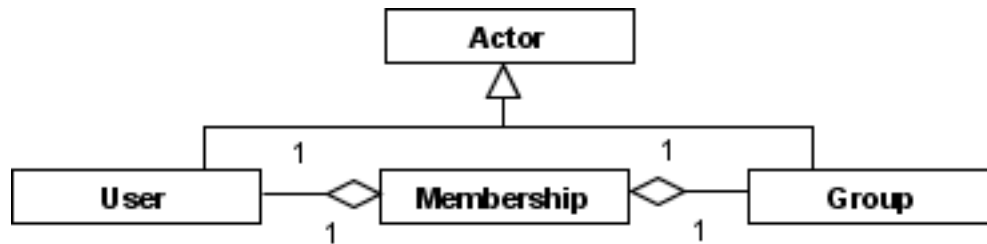
flowId identifies on which flow the authenticated user wants to perform an activity.

attributeValues, transitionName & relations @see [starting a process instance](#)

1.3. The organisation component

The organisation component centralizes all access to organisational information. This includes users, groups and memberships. The organisation component is used by the

execution component in a read-only manner. This is the datamodel used by the default implementation of the organisation component. To couple NetBpm to your own users-database or LDAP system, it is sufficient to switch the organisation component implementation with your own customized version.



The organisational datamodel used in NetBpm

The organisational datamodel used in NetBpm

The NetBpm execution component needs user and group information to assign activities. The execution component retrieves all organisational data through the organisation component. This allows easy configurability to a custom enterprise environment. An company can couple NetBpm to its own infrastructure by providing a custom organisation-component-implementation.

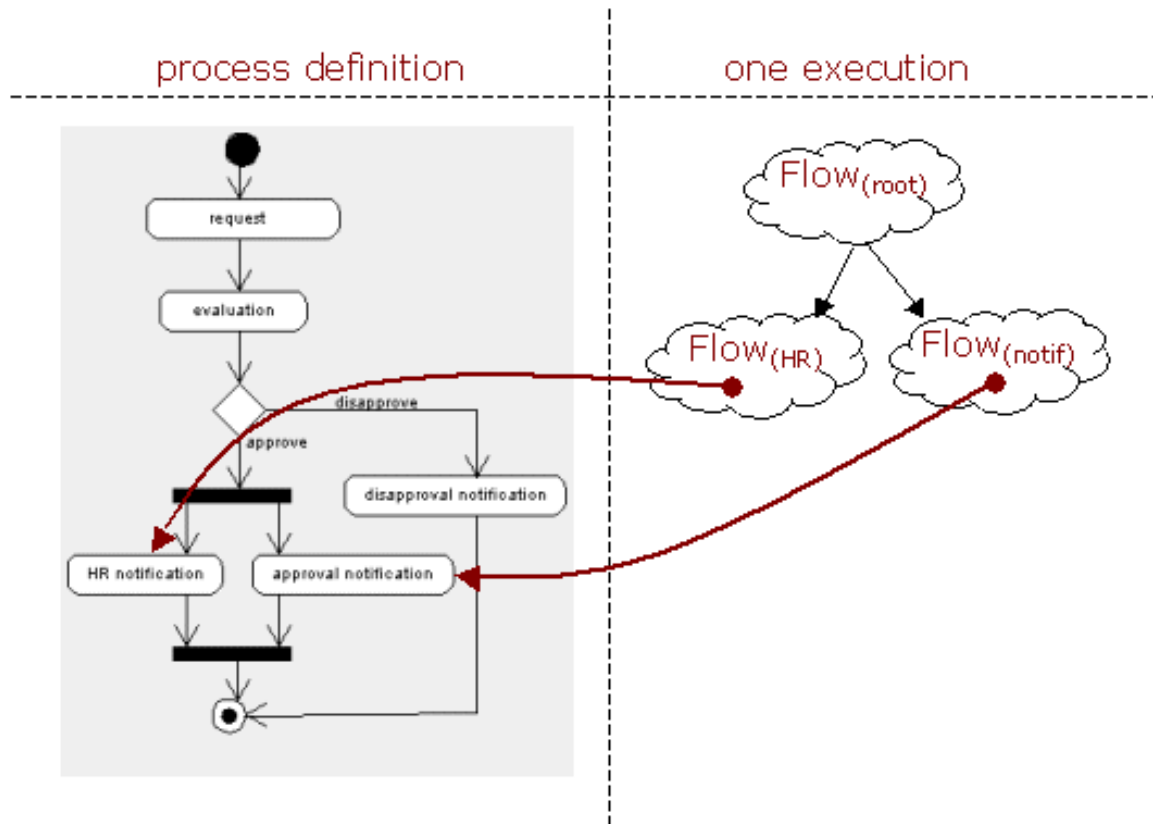
1.4. The log component

1.5. The scheduler component

1.6. The admin component

2. State

A flow represents a 'thread of execution' of activity-states. A process instance represents the complete state of one execution of a process definition. As you can see in the image, the state of a process instance is represented as a hierarchical tree of flows.



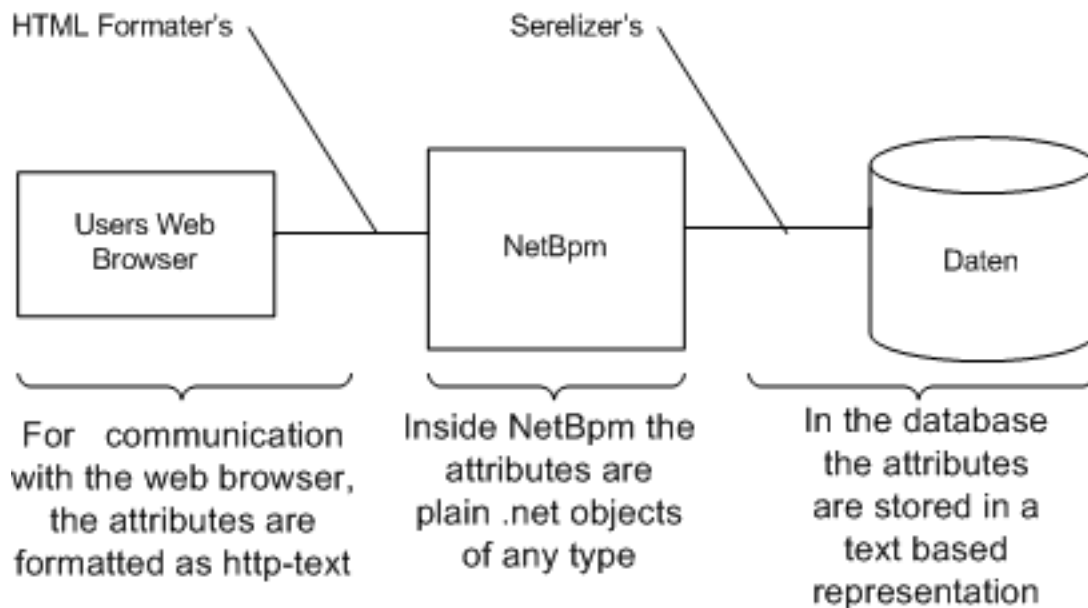
The state of a process instance

The state of a process instance

3. Attributes

3.1. Serialization

Attributes specify process instance variables. An attribute-instance is an instance of an attribute for one process execution. The attributes of a process instance can be seen as a Map that contains process execution related information and that is maintained for the complete execution of the process instance.



The serialization process of attributes

The serialization process of attributes

3.2. Scope

Note that the vision on attributes as explained above is little bit simplified : to be precise, attribute-instances are associated to flows. For each attributes defined in the process definition, an attribute instance is attached to the root-flow. For each attribute defined in a concurrent block, an attribute instance is created for each flow that is forked. Attribute-instances are accessed relative to a flow. A flow is able to 'see' all its attribute-instances plus all attribute-instances of all its parents.

4. The delegation principle

NetBpm is designed as a generic process execution engine. This means that only the basic structure of executing business processes is done by the core NetBpm execution engine. All variable logic is delegated to a specific set of interfaces. These are called the delegation-interfaces. The process definition contains the associations of which delegation implementation to use for every occasion.

The association of delegation-implementations to the process is done in the process definition. To allows for the biggest flexibility, the process developer has to options for every association of a delegation implementation in the process definition :

1. Use one of the default-implementations, already available in the NetBpm-engine.
2. Use a custom implementation which can be included in the process definition as a .NET-assembly.

4.1. The delegation interfaces

The next part will give a short description of each of the delegation interfaces.

4.1.1. Action

An action allows two way communication between the information stored in a process instance and all IT-systems within an organisation. The process-event on which an Action is executed is specified in the process definition. Actions can be seen a kind of listener-interface for process execution events.

4.1.2. DecisionHandler

When execution arrives in an activity-state, an AssignmentHandler has to determine which person or group will be responsible for performing the activity-state.

4.1.3. JoinHandler

Upon arrival of a flow in a join, a JoinHandler decides whether or not the parent flow has to be reactivated

4.1.4. ForkHandler

Upon execution of a fork, a ForkHandler decides which of the leaving transitions will be activated. A ForkHandler also controls the multiplicity of each transition, meaning that multiple flows can be forked for the same transition.

4.1.5. Serializer

This delegation interface is used to convert attribute-values, which are plain .NET-objects to and from text-format. The text-format is used to store attribute-values in the database.

4.1.6. HtmlFormatter

The HtmlFormatter is used to present the HTML-inputcontrol for an attribute and for the parsing of the value, returned by the browser for this HTML-inputcontrol.

4.1.7. ProcessInvocationHandler

A ProcessInvocationHandler handles the spawning and collection of data from a subprocess in an process-state.

5. ProcessInvocationHandler

NetBpm stores all it information in a database. All information of the process definitions (as parsed from the process archives) is stored in the NetBpm-definition tables. The run-time information, which contains the state and information of the execution of process instances is stored in the execution tables. The logging information is stored in the logging tables The security is stored in the organisation tables.

To generate the database creation DDL for your database, see Generating the database in the developers manual.

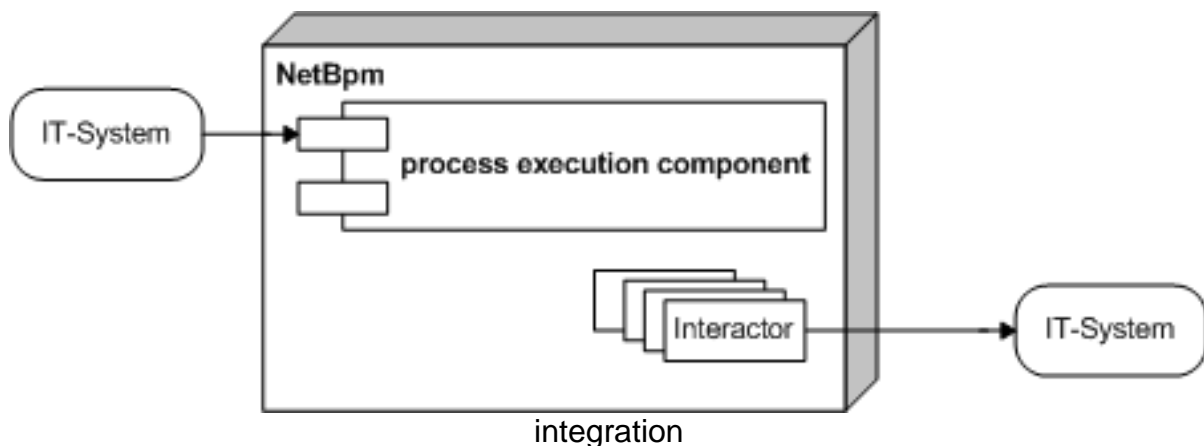
FIXME (pbolle):

describe the generation of the database

6. NetBpm as integration platform

6.1. Process initiated interactio

An Interaction is a delegation interface. An Interaction can be associated with any event during process execution. In the Interaction the developer has access to all the resources that are available within the container like databases through ADO, ... On the other hand, the Interaction-developer has read and write access to the process related data (= attributes).



6.2. Integration initiated by an external application

If an application wants to interact with the execution of processes, it just has to connect to the [execution component's interface](#). The container allows for very easy development of all sorts of connectors which make the interface of the execution component available through various kinds of protocols like : web-service, tcp-socket, http-request, ... In time, all these connectors will be available in the standards NetBpm package. If you develop such a connector, please share it with the community.

7. Exception handling

NetBpm is used as an integration platform. So, when a process is being executed there can be a lot of dependencies to the company's other IT infrastructure. If one of these dependancies causes a failure of the execution of a process, there are three options :

1. the failure is ignored
2. the failure is logged (default)
3. the failure causes a rollback

In case of a rollback, the state of the process instance will be rolled back to the state before the activity was performed. The call to the NetBpm execution interface can be made part of a larger transaction, so that the complete transition is roll-backed in case of a process exception.