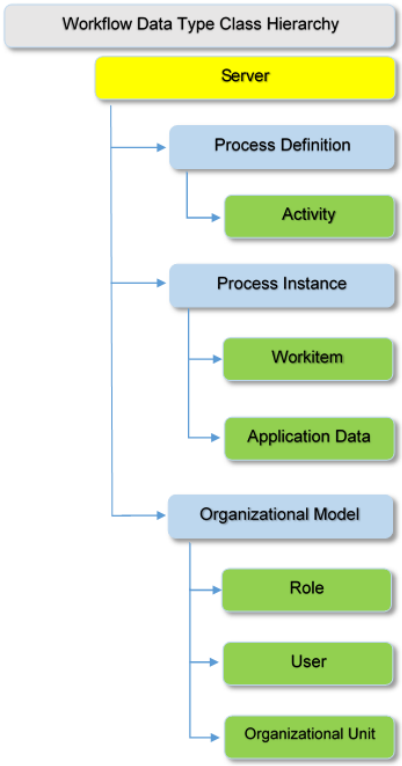
Advanced Workflow API

As mentioned in the previous video (with was Initiation of a process from a GeneXus object, using the Workflow API), most of the API is organized in a hierarchy, with the Server at the top. That means the Server represents the entrypoint to all the features that we have seen so far, including the models, the execution of those models and the user information of those people that execute them. In this particular video, we will be focusing on this object, and the features it enable us to include in our GeneXus code.



The **WorkflowServer** has a series of properties that allow us to get information about the execution environment. Those include:

- The Current Work Session of the logged user.
- The user information and
- The global settings for the entire Server.

Besides those properties, the Server object gives us access to a series of methods and functionalities, which include:

- **Connect and disconnect** into a GXflow session, provided the user name and password. This session is the one the rest of the code in GeneXus will use to execute, and the user will be the one returned in Server.CurrentUser, so some functionalities described next might fail if the current user does not have the proper privileges.

- **List all process definitions** given a series of filters, in a particular order, or directly accessing a single process definition by using its id or name.

- **List all activities** in a process, using some extra filters, or directly accessing one using its id or name.

- **List all instances** (or executions) of workflow processes using some filters, or getting access to a single one using its id.

- **List all elemental workitems** using some filters, or getting access to a single one.

- **Get access to all calendars** defined in a Server, with its holidays and work days definitions, as well as getting Access to a single one if we know its id or name

- **Get access to all nodes in the processes**, that is, obtain information about every activity, every condition, event node and the links between them.

- **Getting** access to the **organizational model** defined in the Server

- **Getting** access to the **events repository**, in order to access the log information of the Server.

- **Getting** access to the **content repository**, in order to access the content of documents in the Server.

For many of those features, the filter data type is an important part of the proper use of the functionality. The object is a superset of all possible filters that might be interesting to us to include when accessing a series of instances, workitems or such objects. Without it, we would be forced to filter in the application on the complete collection, which can be extremely ineffective in production environments that often work with hundreds of processes and millions of workitems.

Not all filters apply to all functions, but it is still useful to know which ones apply where in order to maximize our performance in the use of the API. It is also important to know, however, that using a filter that does not apply to a feature does not trigger an error.

- **The User filter** applies to every entity regulated by the presence of a user, like assigned workitems, triggered events or document custodians
- **The Role filter** applies to entities regulated by roles, like candidates for an activity or users with some role assigned
- **The OrganizationalUnit filter** applies to entities regulated by them, like users or roles
- **The ProcessDefinition filter** applies to all entities that include a ProcessDefinition. Those include workitems, process instances, activities and candidate roles.
- **The ProcessInstance filter** applies to entities that are defined by an instance, which include documents, workitems and assigned users.
- **The Subject filter** applies to the subject of the instances and the workitems. This filter is case sensitive.
- **The Activity filter** applies to workitems and candidate roles of that activity.
- **The From and To** filters define a lower and upper limit for the creation date of documents, events, process instances and workitems

- **The Name filter** applies to the descriptive name of documents types, the title of documents and the names of roles, restrictions, users, tasks and proceses definitions. All this filters are case sensitive.

- **Start and Limit** are used to improve the performance of getting several items, to use on functionalities like reporting or paging. Start specifies the offset of the dataset retrieved, while limit specifies the amount of objects returned. Most modern DBMS support this functionality at the level of the Server, instead of the driver, improving bandwidth and transfer time with the database, although they might be some restrictions to its use. For example, SQL Server supports this functionality since 2012, but it requires an order to be specified.

- **State** includes objects that are affected by the workflow state machine, so it can be used to filter process definitions, instances, workitems and documents. This filter is quite complex, since it has a series of posible values, and not all of the apply to the different object types, so let's dig down a Little further:

When used with process definitions, they can be either enabled (meaning, new instances can be created) or disabled (so no new instances can be created)

WorkflowProcessDefinitionState
ENABLED
DISABLED

When used with processes instances, they can be:

OPEN_RUNING: When an instance is created and executed normally.

OPEN_NOTRUNING_SUSPENDED: When the functionality of Suspend on the instance is executed, the state is changed to NOTRUNING_SUSPENDED. Also, all subprocesses and active workitems are changed to suspended, to ensure no part of the instance remains active.

CLOSED_COMPLETED: This is the state used when an instance is completed normally.

CLOSED_ABORTED: Similar to suspended, when the functionality of Abort the instance is executed, said instance and all its components are set to this state.

CLOSED_TERMINATED: This state is used when the instance is terminated unexpectedly, particularly through a deadline.

When used on a workitem, the state filter can be:

OPEN_ACTIVE_READY: When the workitem is created

OPEN_ACTIVE_ASSIGNED: When it has been assigned or taken by a user

OPEN_ACTIVE_INPROCESS: While the workitem is in execution, that is, after it was taken but before it was sent.

CLOSED_COMPLETED: When the workitem was sent back and considered completed.

Returning to the filters:

- **The** filters **CreatedFrom** and **CreatedBy** work similarly to From and To. In fact, From/To are maintained for compatibility proposes.

- **The** filters **EndedFrom** and **EndedBy** work similarly to the previous ones, but for objects that have a termination date, like workitems and instances. It is important to detail that instances that are currently open will be cut down by these filters if specified. In case you are interested in currently open instances, you should use the State filter instead.

- **Search** can be used to filter a combination of different attributes in different entities: for workitems it filters by its subject, its name, its process' name, its task's name or its user's name; for users, it filters by code, name or e-mail; for restrictions, it filters by code or value; for processes, it filters by name and description; for instances, it filters by its subject, its process' name or its user's name and for documents, its title or its responsible.

- **The Workflow priority** filter can be used to filter workitems and instances by their set priority, the options being:

- **High priority**
- **Normal priority** and
- **Low priority**

Another object that we have to consider when programming with GeneXus on the API is the workflow context. Generally, objects that have to be used as gateways require the following **parameters**. The Process Definition, the process instance and the workitem, to identify where in the workflow we are standing.

However, since the latest versions of GeneXus, we have introduced the workflow **context**, which is defined automatically on every transaction and webpanel that is called from workflow and includes:

- **The** process definition that is currently running
- **The** process instance that is currently running
- **And** the current workitem

And also the functionality to **check** the rights of a particular user with a particular activity.

This means that the same code in the same object will behave differently whether it is executed within the **inbox** or as an **external** url.

```
Event Start
    msg(!"Workitem "+&WorkflowContext.Workitem.Id)
EndEvent
```

Inbox

External url

The WorkflowUser data type expands on the properties that define a user in Workflow, which include: a **unique** user name, a **full** name, an **email** (which is optional and not unique), the **Access** level, defining if the user can execute this API on other users or only on itself, a **set** of roles, organizational units and restrictions, an **external** code, which is used to interface with other modules, like the GeneXus Access Manager and **whether** the user is currently connected, blocked, or "Out of office". In case it is out of office, the properties also include its **substitute**.

Besides those properties, the data type includes a lot of functionality. You can get the user's **current** worklist, or the **list** of processes the user has rights to start, using the filters described before. Provided your user has enough Access level, you can also get the **roles** defined within the OrganizationalUnit that the user has. You can **assign** or remove roles, organization units or restrictions from the user.

As an administrator, you can also **block** or enable a user, as well as set it as "**Out** of Office", in which case you may also select another user as a substitute.

Finally, you can force to **rebuild** the user worklist. Doing so will reapply all rules related to candidate roles for the workitems the user may have, considering the proper restrictions and organizational units in order to, in essence, refresh the list of items a user may take and execute. The Server instalation is configured by default to perform these calculations for any change in the user rights, as well as movements in the state machine, automatically, but since they can be resource intensive and time consuming in environments with dozens of roles, hundreds of users and millions of workitems, you are given the option to disable the automatic rebuild and manually trigger it under certain conditions (for example, when a user logs in)

The last set of data type we would like to discuss are the roles, restrictions and organizational units. Even when they are conceptually different and have different use cases, they share the characteristic that they are all modifiers to define what a user can or can't do.

As such, they all have a **name or value**, and also the functionalities to **add** and **remove** users to their metadata.