

# **WebSphere® Application Server Enterprise Process Choreographer**

## **Staff Resolution Parameter Reference**

Applicable to release: 5.0  
Document version: 1.0.1  
Publish date: March 28, 2003  
Author: Kurt Lind ( [klind@de.ibm.com](mailto:klind@de.ibm.com) )



---

## Contents

---

<b>1</b>	<b>Preface</b>	<b>1</b>
<b>2</b>	<b>Staff Query Verb Reference</b>	<b>3</b>
2.1	Predefined Staff Query Verbs . . . . .	3
2.1.1	Department Members . . . . .	4
2.1.2	Everybody . . . . .	4
2.1.3	Group Members . . . . .	5
2.1.4	Group Search . . . . .	6
2.1.5	Manager of Employee . . . . .	7
2.1.6	Manager of Employee by user ID . . . . .	8
2.1.7	Native Query . . . . .	8
2.1.8	Nobody . . . . .	10
2.1.9	Person Search . . . . .	10
2.1.10	Role Members . . . . .	13
2.1.11	Users . . . . .	14
2.1.12	Users by user ID . . . . .	14
2.2	Staff Query Verb Syntax . . . . .	15
2.3	Parameterized Verb Syntax . . . . .	17
<b>3</b>	<b>Staff Query Reference</b>	<b>19</b>
3.1	Common Syntax Elements . . . . .	20
3.1.1	Query Frame . . . . .	20
3.1.2	Simple Queries . . . . .	20
3.1.3	Context Variables . . . . .	21
3.2	System Staff Resolution Plug-in . . . . .	22
3.3	User Registry Staff Resolution Plug-in . . . . .	22

3.3.1	User Query . . . . .	23
3.3.2	Users Of Group Query . . . . .	23
3.3.3	Search Query . . . . .	23
3.3.4	Samples . . . . .	24
3.4	LDAP Staff Resolution Plug-in . . . . .	24
3.4.1	User Query . . . . .	25
3.4.2	Users Of Group Query . . . . .	25
3.4.3	Search Query . . . . .	25
3.4.4	Attribute Evaluation Specifications . . . . .	26
3.4.5	Samples for “user”, “usersOfGroup” and “search” Queries . . . . .	28
3.4.6	Queries of type “intermediateResult” . . . . .	31
<b>4</b>	<b>Staff Plug-in Provider Parameter Reference</b>	<b>33</b>
4.1	System Staff Resolution Plug-in . . . . .	33
4.2	User Registry Staff Resolution Plug-in . . . . .	33
4.3	LDAP Staff Resolution Plug-in . . . . .	34
<b>A</b>	<b>XML schemas</b>	<b>35</b>
A.1	Staff Query Verb Set . . . . .	35
A.2	Parameterized Verb . . . . .	37
A.3	Staff Resolution Plug-ins . . . . .	37
A.3.1	System Staff Resolution Plug-in . . . . .	38
A.3.2	User Registry Staff Resolution Plug-in . . . . .	39
A.3.3	LDAP Staff Resolution Plug-in . . . . .	40
<b>B</b>	<b>References</b>	<b>45</b>
<b>C</b>	<b>Trademark attributions and copyrights</b>	<b>47</b>

# CHAPTER 1

---

## Preface

---

### **Who is this document for?**

This document is intended for application developers and solution architects. It provides a detailed description, examples, and reference material for the IBM® WebSphere® Application Server Enterprise Process Choreographer (WPC) staff support.

**Chapter 2** describes the predefined staff query verb set and the structure of a verb set XML document. This reference is helpful when modeling business processes or customizing the staff query verb set.

**Chapter 3** provides a detailed description of the staff query XML syntax for the specialized staff resolution plug-ins and gives some examples of staff queries. This information is useful when customizing the staff verb mapping XSLT files.

**Chapter 4** describes the configuration parameters that are supported by the staff resolution plug-ins. This reference information is needed to configure the staff plug-in provider in the WebSphere Application Server Administration Console.

This document provides comprehensive reference documentation for the staff support of WPC, it does not fully describe WPC, nor the WPC staff support. For a description of the WPC concepts and architecture, see [1], the WPC programming model is described in [2]. The staff resolution architecture is documented in [3] and the staff support programming model in [4]. For more information about these documents, see section B on page 45.

### **About the Author**

The author of this document, Kurt Lind (klind@de.ibm.com) works in the development team for WebSphere Application Server Enterprise Process Choreographer. He is responsible for the WPC staff resolution.

## Summary of changes

### From Version 1.0.0 to Version 1.0.1:

- Minor changes in the verb set XML schema file in section A.1 on page 35.
- Update of the “References” section. See section B on page 45.
- Addition of a new “Trademark attributions and copyrights” section. See section C on page 47.

## CHAPTER 2

---

### Staff Query Verb Reference

---

When you use WebSphere Studio Application Developer Integration Edition to model business processes that make use of staff activities, the Process Editor provides you with a predefined set of staff query verbs. These verbs are abstract query templates, which can be used to define concrete staff queries against a staff repository, such as LDAP. The process modeller populates the abstract staff query parameters with values, which the Process Editor uses to generate the parameterized verb. This becomes the input for the XSLT mapping that generates the actual executable staff query, which is specialized for a specific staff resolution plug-in, and thus staff repository.

To gain a better understanding of staff query verbs and their usage, refer to [3] and [4] in section B on page 45.

This chapter describes the set of predefined staff query verbs, and the syntax of the XML file that contains the verb set definition. The syntax is of interest for those who want to customize or extend the predefined verb set. Additionally, this chapter describes the XML format of the parameterized verb, as generated by the Process Editor. This syntax is important for those who want to customize the XSLT mapping files.

#### **2.1 Predefined Staff Query Verbs**

The set of predefined staff query verbs available in the Process Editor, is delivered in the file “VerbSet.xml” and can be used directly, without further customization.

Depending on the staff resolution plug-in and XSLT mapping file used, not all these verbs may be supported. The “Manager of Employee” verb, for example, is not available when using the User Registry or the System plug-in.

The following subsections give a detailed description of the predefined staff verbs delivered with WPC that are available when modeling processes in the Process Editor.

### 2.1.1 Department Members

This verb allows you to define a query to retrieve the members of a department. The retrieved users will belong to any of the specified departments (DepartmentName or AlternativeDepartmentName1 or AlternativeDepartmentName2).

This verb is supported by the following plug-ins:

- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

The “Domain” parameter is not supported.

#### Parameters:

- **DepartmentName**  
**Usage:** mandatory  
**Type:** xsd:string  
Specifies the department name of the users to be retrieved.
- **IncludeNestedDepartments**  
**Usage:** mandatory  
**Type:** xsd:boolean  
Specifies whether nested departments will be considered for this query.
- **Domain**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the domain to which the department belongs. The Domain parameter allows you to limit the query to a subset of the directory.
- **AlternativeDepartmentName1**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative department name to which the users can belong.
- **AlternativeDepartmentName2**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative department name to which the users can belong.

### 2.1.2 Everybody

This verb allows you to assign a work item to every user authenticated by the Web-Sphere Application Server.

This verb is supported by the following plug-ins:

- **System**

- **User Registry**
- **LDAP**

**Parameters:** This verb has no parameters.

### 2.1.3 Group Members

This verb allows you to define a query to retrieve the members of a group. The retrieved users will belong to any of the specified groups (GroupName or AlternativeGroupName1 or AlternativeGroupName2).

This verb is supported by the following plug-ins:

- **User Registry** - The “Domain” and “IncludeSubgroups” parameters are not supported.
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.  
The “Domain” parameter is not supported.

**Parameters:**

- **GroupName**  
**Usage:** mandatory  
**Type:** xsd:string  
Specifies the group name of the users to be retrieved.
- **IncludeSubgroups**  
**Usage:** mandatory  
**Type:** xsd:boolean  
Specifies whether the nested subgroups will be considered for this query.
- **Domain**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the domain to which the group belongs. The Domain parameter allows you to limit the query to a subset of the directory.
- **AlternativeGroupName1**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative group name to which the users can belong.
- **AlternativeGroupName2**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative group name to which the users can belong.

## 2.1.4 Group Search

This verb allows you to search for a group based on an attribute match, and to retrieve the members of the group.

This verb is supported by the following plug-ins:

- **User Registry** - Only the “GroupID” parameter is supported.
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

### Parameters:

- **GroupID**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group ID of the users to be retrieved.
- **Type**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group type of the users to be retrieved.
- **IndustryType**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group industry type of the users to be retrieved.
- **BusinessType**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group business type of the users to be retrieved.
- **GeographicLocation**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group geographic location of the users to be retrieved.
- **Affiliates**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group affiliates of the users to be retrieved.
- **DisplayName**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group display name of the users to be retrieved.

- **Secretary**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group secretary of the users to be retrieved.
- **Assistant**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group assistant of the users to be retrieved.
- **Manager**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group manager of the users to be retrieved.
- **BusinessCategory**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group business category of the users to be retrieved.
- **ParentCompany**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the group parent company of the users to be retrieved.

### 2.1.5 Manager of Employee

This verb allows you to retrieve the manager of a person, given the person's name.

This verb is supported by the following plug-ins:

- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.  
The "Domain" parameter is not supported.

#### Parameters:

- **EmployeeName**  
**Usage:** mandatory  
**Type:** xsd:string  
Specifies the name of the employee whose manager will be retrieved.
- **Domain**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the domain to which the group belongs. The Domain parameter allows you to limit the query to a subset of the directory.

### 2.1.6 Manager of Employee by user ID

This verb allows you to retrieve the manager of a person, given the person's user ID. It is useful in combination with context queries.

This verb is supported by the following plug-ins:

- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

The "Domain" parameter is not supported.

#### Parameters:

- **EmployeeUserID**

**Usage:** mandatory

**Type:** xsd:string

Specifies the user ID of the employee whose manager will be retrieved. This parameter allows the use of context variables like "%wf:process.starter%".

- **Domain**

**Usage:** optional

**Type:** xsd:string

Specifies the domain to which the group belongs. The Domain parameter allows you to limit the query to a subset of the directory.

### 2.1.7 Native Query

This verb allows you to define a native query based on directory-specific parameters.

This verb is supported by the following plug-ins:

- **User Registry**
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

#### Parameters:

- **QueryTemplate**

**Usage:** mandatory

**Type:** xsd:string

Specifies the query template to be used by the native query. The User Registry mapping XSLT file shipped with the product supports the templates "search", "user" and "usersOfGroup". The LDAP mapping XSLT file shipped with the product supports the templates "search", "user" and "usersOfGroup".

- **Query**

**Usage:** mandatory

**Type:** xsd:string

Specifies the actual query. This parameter allows the use of context variables like “%wf:process.starter%”. Depending on the plug-in and query template used, it will have different meanings.

The following are the meanings for the default mapping XSLT files shipped with the product:

**User Registry:**

- **Template “search”** - search pattern
- **Template “user”** - user name
- **Template “usersOfGroup”** - group name

**LDAP:**

- **Template “search”** - search filter
- **Template “user”** - user dn
- **Template “usersOfGroup”** - group dn

● **AdditionalParameter1**

**Usage:** optional

**Type:** xsd:string

Allows you to specify an additional parameter. Depending on the plug-in and query template used, it will have different meanings.

The following are the meanings for the default mapping XSLT files shipped with the product:

**User Registry:**

- **Template “search”** - search type
- **Template “user”** - not supported
- **Template “usersOfGroup”** - not supported

**LDAP:**

- **Template “search”** - parameter “recursive”. Allowed values are “yes” and “no”.
- **Template “user”** - not supported
- **Template “usersOfGroup”** - parameter “recursive”. Allowed values are “yes” and “no”.

● **AdditionalParameter2**

**Usage:** optional

**Type:** xsd:string

Allows you to specify an additional parameter. Depending on the plug-in and query template used, it will have different meanings. When using the default mapping XSLT files shipped with the product, it can be only used to specify the parameter “baseDN” for LDAP searches.

- **AdditionalParameter3**

**Usage:** optional

**Type:** xsd:string

Allows you to specify an additional parameter. Depending on the plug-in and query template used, it will have different meanings. When using the default mapping XSLT files shipped with the product, it is not supported.

- **AdditionalParameter4**

**Usage:** optional

**Type:** xsd:string

Allows you to specify an additional parameter. Depending on the plug-in and query template used, it will have different meanings. When using the default mapping XSLT files shipped with the product, it is not supported.

- **AdditionalParameter5**

**Usage:** optional

**Type:** xsd:string

Allows you to specify an additional parameter. Depending on the plug-in and query template used, it will have different meanings. When using the default mapping XSLT files shipped with the product, it is not supported.

### 2.1.8 Nobody

This verb denies normal users access to the work item - only the process administrator and the WPC system administrator have access to it.

This verb is supported by the following plug-ins:

- **System**
- **User Registry**
- **LDAP**

**Parameters:** This verb has no parameters.

### 2.1.9 Person Search

This verb allows you to search for people based on an attribute match.

This verb is supported by the following plug-ins:

- **User Registry** Only the “UserID” parameter is supported.
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

**Parameters:**

- **UserID**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the user ID of the people to be retrieved.
- **Profile**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the profile of the people to be retrieved.
- **LastName**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the last name of the people to be retrieved.
- **FirstName**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the first name of the people to be retrieved.
- **MiddleName**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the middle name of the people to be retrieved.
- **Email**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the e mail address of the people to be retrieved.
- **Company**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the company number of the people to be retrieved.
- **DisplayName**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the display name of the people to be retrieved.
- **Secretary**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the secretary of the people to be retrieved.

- **Assistant**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the assistant of the people to be retrieved.
- **Manager**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the manager of the people to be retrieved.
- **Department**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the department of the people to be retrieved.
- **EmployeeID**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the employee ID of the people to be retrieved.
- **Department**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the tax payer ID of the people to be retrieved.
- **Phone**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the phone number of the people to be retrieved.
- **Fax**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the fax number of the people to be retrieved.
- **Gender**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the gender of the people to be retrieved.
- **Timezone**  
**Usage:** optional  
**Type:** xsd:string  
Specifies the timezone of the people to be retrieved.

- **PreferredLanguage**

**Usage:** optional

**Type:** xsd:string

Specifies the preferred language of the people to be retrieved.

### 2.1.10 Role Members

This verb allows you to define a query to retrieve the users associated with a business process role. The retrieved users will belong to any of the specified roles (RoleName or AlternativeRoleName1 or AlternativeRoleName2).

This verb is supported by the following plug-ins:

- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

The “Domain” parameter is not supported.

**Parameters:**

- **RoleName**

**Usage:** mandatory

**Type:** xsd:string

Specifies the role name of the users to be retrieved.

- **IncludeNestedRoles**

**Usage:** mandatory

**Type:** xsd:boolean

Specifies whether the nested roles will be considered for this query.

- **Domain**

**Usage:** optional

**Type:** xsd:string

Specifies the domain to which the role belongs. The Domain parameter allows you to limit the query to a subset of the directory.

- **AlternativeRoleName1**

**Usage:** optional

**Type:** xsd:string

Specifies an alternative role name to which the users can belong.

- **AlternativeRoleName2**

**Usage:** optional

**Type:** xsd:string

Specifies an alternative role name to which the users can belong.

### 2.1.11 Users

This verb allows you to define a staff query for a user who is known by name. It is not recommended to hard code user names in process templates, this verb is most useful for test purposes.

This verb is supported by the following plug-ins:

- **System**
- **User Registry**
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

#### Parameters:

- **Name**  
**Usage:** mandatory  
**Type:** xsd:string  
Specifies the name of the user to be retrieved.
- **AlternativeName1**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative user name. This will allow you to retrieve more than one user.
- **AlternativeName2**  
**Usage:** optional  
**Type:** xsd:string  
Specifies an alternative user name. This will allow you to retrieve more than one user.

### 2.1.12 Users by user ID

This verb allows you to define a staff query for a user who is known by his or her user ID. Even though it is not recommended to hard code user names in process templates, this verb is useful in combination with context queries, for example:

```
Users by user ID [UserID='%wf:process.starter%']
```

This verb is supported by the following plug-ins:

- **System**
- **User Registry**
- **LDAP** - Depending on the LDAP schema used, you might have to customize the default mapping XSLT file.

**Parameters:**

- **UserID**

**Usage:** mandatory

**Type:** xsd:string

Specifies the user ID of the user to be retrieved. This parameter allows the use of context variables like “%wf:process.starter%”.

- **AlternativeID1**

**Usage:** optional

**Type:** xsd:string

Specifies an alternative user ID. This will allow you to retrieve more than one user. This parameter allows the use of context variables like “%wf:process.starter%”.

- **AlternativeID2**

**Usage:** optional

**Type:** xsd:string

Specifies an alternative user ID. This will allow you to retrieve more than one user. This parameter allows the use of context variables like “%wf:process.starter%”.

## 2.2 Staff Query Verb Syntax

If the set of predefined staff query verbs do not satisfy your requirements for a specific query, you can modify the existing verbs or add new verbs to the verb set. This section explains the structure of the verb set file and the syntax of the staff query verbs.

The structure of the verb set XML file is simple:

```
<?xml version="1.0" encoding="UTF-8"?>
<vs:VerbSet xmlns:vs=
  "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/verbset">
  <vs:Description>verb set description</vs:Description>
  verb definition 1+
</vs:VerbSet>
```

It contains the usual XML prologue, the namespace declaration, the verb set description and the actual verb definitions. The syntax of the verb definitions is explained below.

The verb definition has the following syntax:

```
<vs:DefineVerb name="verb name">
  <vs:Description>verb description</vs:Description>
  <vs:Mandatory>
    parameter definition *
  </vs:Mandatory>
  <vs:Optional>
    parameter definition *
  </vs:Optional>
</vs:DefineVerb>
```

The root element has the attribute “name”, which identifies the verb. It is followed by the verb description and the set of mandatory parameters. This set is followed then by the set of optional parameters. There can be zero or more parameters in each of the mandatory and optional parameter sets.

```
<vs:Parameter>
  <vs:Name>parameter name</vs:Name>
  <vs:Type>parameter type</vs:Type>
  <vs:Hint>hint text</vs:Hint> *
</vs:Parameter>
```

The parameter definition is identical for optional and mandatory parameters. It contains the parameter name, its data type and a number of optional hints.

The parameter type and value space corresponds to the XML schema data types, it can be one of the following:

- boolean
- date
- dateTime
- decimal
- double
- duration
- float
- gDay
- gMonth
- gMonthDay
- gYear
- gYearMonth
- integer
- string
- time

**Note:** The parameter values entered by modellers using the WebSphere Studio Application Developer Integration Edition Process Editor are not validated, except for the type boolean.

The hint texts will be displayed in the Process Editor inside a drop-down box in the Staff Query Definition dialog.

For a formal specification of the verb set XML syntax, see the corresponding XML schema file in the appendix, section A.1 on page 35.

The following is a simple example of a verb set file:

```

<?xml version="1.0" encoding="UTF-8"?>
<vs:VerbSet xmlns:vs=
  "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/verbset">
  <vs:Description>Default Verbset</vs:Description>

  <vs:DefineVerb name='Nobody'>
    <vs:Description>Assigns no normal users.
Only the process and WPC system administrators have access.
Supported by sample XSLT files for:
- LDAP
- User Registry
- System
    </vs:Description>
    <vs:Mandatory></vs:Mandatory>
    <vs:Optional></vs:Optional>
  </vs:DefineVerb>

  <vs:DefineVerb name='Users by user ID'>
    <vs:Description>Assigns users, given their user ID.
Supported by sample XSLT files for:
- LDAP
- User Registry
- System
    </vs:Description>
    <vs:Mandatory>
      <vs:Parameter>
        <vs:Name>UserID</vs:Name>
        <vs:Type>xsd:string</vs:Type>
        <vs:Hint>%wf:process.starter%</vs:Hint>
        <vs:Hint>%wf:process.administrators%</vs:Hint>
        <vs:Hint>
          %wf:activity(-enter activity name-).readers%
        </vs:Hint>
      </vs:Parameter>
    </vs:Mandatory>
    <vs:Optional>
      <vs:Parameter>
        <vs:Name>AlternativeID1</vs:Name>
        <vs:Type>xsd:string</vs:Type>
      </vs:Parameter>
      <vs:Parameter>
        <vs:Name>AlternativeID2</vs:Name>
        <vs:Type>xsd:string</vs:Type>
      </vs:Parameter>
    </vs:Optional>
  </vs:DefineVerb>
</vs:VerbSet>

```

## 2.3 Parameterized Verb Syntax

When a staff query verb is selected and its parameter values are set in the Process Editor, the result is another XML document snippet, with content similar to the abstract staff query verbs. This snippet is inserted into the business process model specification contained in the process EAR (enterprise archive) file to be deployed. During deployment the snippet becomes the input for the XSL transformation initiated by the Staff Support Service, which generates the staff resolution plug-in specific query syntax for this verb.

This section explains the parameterized verb syntax.

The structure of the parameterized verb XML document is very simple, it is a simplified form of the abstract staff query verbs:

```
<?xml version="1.0" encoding="UTF-8"?>
<staff:verb xmlns:staff=
  "http://www.ibm.com/schemas/workflow/wswf/plugins/staff">
  <staff:name>verb name</staff:name>
  <staff:id>verb ID</staff:id>
  <staff:parameter id="parameter ID">
    parameter value
  </staff:parameter> 1+
</staff:verb>
```

It contains the usual XML prologue, the namespace declaration, the verb name, the mandatory verb ID and the list of parameters. The verb ID identifies the verb, and the verb name is an optional value that is usually identical to the ID.

At least one parameter must be specified. It is identified by its attribute "id". The element text content forms the parameter value.

For a formal specification of the parameterized verb XML syntax, see the corresponding XML schema file in the appendix, section A.2 on page 37.

The following is a simple example of a parameterized verb document:

```
<?xml version="1.0" encoding="UTF-8"?>
<staff:verb xmlns:staff=
  "http://www.ibm.com/schemas/workflow/wswf/plugins/staff">
  <staff:name>Users by user ID</staff:name>
  <staff:id>Users by user ID</staff:id>
  <staff:parameter id="UserID">
    %wf:process.starter%
  </staff:parameter>
  <staff:parameter id="AlternativeID1">John</staff:parameter>
</staff:verb>
```

## CHAPTER 3

---

### Staff Query Reference

---

Staff queries are templates that define how to retrieve the list of users authorized for a certain business process work item. Starting as abstract query templates (staff verbs) in the WebSphere Studio Application Developer Integration Edition Process Editor, they can be used to model a business process. After some transformations during modeling and deployment, they end as a set of concrete queries against a staff repository, executed at runtime.

The initial staff query verbs visible in the Process Editor, are abstract templates that are not executable as query against a concrete directory. Even the parameterized verbs included in the process models are not yet executable. First they have to be translated into executable queries using an XSL transformation. The result of such a transformation (mapping) can be executed by the specialized staff resolution plug-ins that provide access to specific directories such as LDAP or the User Registry. At run-time, the plug-in will execute this query by invoking the staff repository's APIs and creating the list of authorized user IDs for the corresponding process work item.

When a process is deployed, the JNDI name of a staff plug-in provider configuration is provided as a deployment descriptor variable. All staff query verbs belonging to a process template must use the same provider configuration. During the process deployment, the Staff Support Service (SSS) is invoked to deploy a particular staff query. It retrieves the staff plug-in provider configuration with the corresponding JNDI name, and on behalf of the XSLT verb mapping file and the staff resolution plug-in defined in the configuration, the SSS converts the parameterized staff query verb into a query syntax that can be executed by the specialized staff resolution plug-in.

For a better understanding of the deployment process, refer to [3] and [4] in section B on page 45. These documents fully describe the staff support architecture and programming model.

This chapter describes the syntax of the concrete and executable staff queries that will be generated by the XSL transformation. You will need this information when customizing or creating new verb mapping XSLT files.

## 3.1 Common Syntax Elements

All staff resolution plug-ins shipped with WebSphere Application Server Enterprise Process Choreographer (WPC) have a common set of syntax elements. These elements differ only in the XML namespace that they belong to, each plug-in has its own namespace. This section describes the common part of the plug-in syntax.

For a formal specification of the query XML syntax, see the corresponding XML schema files in the appendix, section A.3 on page 37.

### 3.1.1 Query Frame

The frame for all queries is identical:

```
<staff:staffQueries threshold="positive nonzero integer number" >
  staff query element*
</staff:staffQueries>
```

The `<staffQueries>` specification can contain one or more staff query elements - except for `<everybody/>` and `<nobody/>` that allow no further staff query elements. The results of all queries are concatenated, to create the resulting set of user IDs.

All specialized plug-ins support a “threshold” parameter. This parameter allows you to limit the total number of users that are returned by a staff query. If the number of users that are being retrieved is larger than the threshold value, the query is aborted at this point and only the users resolved before the threshold was reached will be returned. Other users that would have qualified for this query will be ignored. In the business process context it is normally not required to assign a large number of users to an activity. If a staff query returns a large number of users (for example, more than 50) it is likely that a query is not defined in a well-contained manner. Independently of the “threshold” parameter many directories impose their own limitations on the size of result sets.

The “threshold” parameter is optional. Its value must be a positive nonzero integer number. The default for the threshold value is 2,147,483,647.

### 3.1.2 Simple Queries

There are several simple staff queries that do not require access to a directory. These queries are supported by every staff resolution plug-in shipped with WPC:

- **Everybody** - every authenticated user.

```
<staff:staffQueries>
  <staff:everybody/>
</staff:staffQueries>
```

**Note** that the `threshold` attribute of the `<staffQueries>` element will be ignored by this query.

**Note** also that beside the `<everybody/>` query no further query (like `<userID>`) is allowed.

- **Nobody** - no normal user. Only the process administrator and the WPC system administrator have access to this work item.

```
<staff:staffQueries>
  <staff:nobody/>
</staff:staffQueries>
```

**Note** that the `threshold` attribute of the `<staffQueries>` element will be ignored by this query.

**Note** also that beside the `<nobody/>` query no further query (like `<userID>`) is allowed.

- **User** - explicit assignment to a specific WebSphere Application Server authenticated user ID.

```
<staff:staffQueries>
  <staff:userID name="valid WebSphere Application Server authenticated user ID" /> 1+
</staff:staffQueries>
```

Even though this query allows you to assign users by name, it is a good practice to avoid hard coding user names in the process design. However, the ability to assign work items to users by name is useful for assignments that are based on context data (for example, the process starter or a person defined in the input message).

The “name” attribute is mandatory. It can contain context variables like `%wf:process.starter%`. The number of multi-value context variables is restricted to one variable per `<staff:userID . . ./>` element. Any expressions containing context variables must evaluate to a valid WebSphere Application Server authenticated user ID.

### 3.1.3 Context Variables

Several staff query elements support context variables for some of their parameters. Context variables are encapsulated in “%” signs and have the following notation: `%parameter name%`, for example, `%wf:process.starter%`.

Single value context variables evaluate to a string that gets replaced in the parameter context, for example, `cn=%wf:process.starter%` evaluates to `cn=Sarah Miller`. Multi-value context variables like `%wf:process.administrators%` evaluate to a string array and the staff query element containing them must be executed many times (corresponding to the array length), each time with a different value from the result array. Thus the number of multi-value context variables must be limited to one per staff query element.

If the variable content is null, then no replacement occurs and the variable name remains in the query string (including the “%” signs). This query will then probably fail.

WPC supports the following context variables:

- **wf:process.starter** is bound to the starter of the current process instance.

- **wf:process.administrators** is bound to the administrators of the current process instance.
- **wf:process.readers** is bound to the readers of the current process instance.
- **wf:activity(name).potentialOwners** is bound to the potential owners of the activity instance with the supplied name.
- **wf:activity(name).owner** is bound to the owner of the activity instance with the supplied name.
- **wf:activity(name).editors** is bound to the editors of the activity instance with the supplied name.
- **wf:activity(name).readers** is bound to the readers of the activity instance with the supplied name.

The variables bound to an activity are only valid after this activity has gone in the ready state, or has been started.

For example: `wf:activity(CollectCreditInfo).potentialOwners` evaluates to the list of potential owners for the activity “CollectCreditInfo”.

## 3.2 System Staff Resolution Plug-in

The System staff resolution plug-in only supports the simple queries described above.

It is not intended to be used in production, since it does not allow real staff queries, only hard coded staff assignments. However, it allows you to write simple staff business process test scenarios, without having to configure and use a staff repository.

The formal definition of the supported staff queries as XML schema can be found in section A.3.1 on page 38. The XML namespace used by the deployment elements of this plug-in is defined as:

```
xmlns:ssys=
"http://www.ibm.com/schemas/workflow/wswf/plugins/staff/system"
```

## 3.3 User Registry Staff Resolution Plug-in

The User Registry staff resolution plug-in allows you to refer to users and groups that are known to the WebSphere Application Server. The implementation is based on the WebSphere Application Server User Registry interface.

**Note** that the deprecated `getUsersForGroup()` method of the UserRegistry interface must be implemented for Custom Registries, in order to be supported by the User Registry staff resolution plug-in.

In addition to the three simple queries described in section 3.1.2 on page 20, this plug-in also supports the following query elements:

- `<sur:user .../>`
- `<sur:usersOfGroup .../>`

- `<sur:search .../>`

The formal definition of the supported staff queries as XML schema can be found in section A.3.2 on page 39. The XML namespace used by the deployment elements of this plug-in is defined as:

```
xmlns:sur=
"http://www.ibm.com/schemas/workflow/wswf/plugins/staff/userregistry"
```

### 3.3.1 User Query

The format to lookup for the user ID of a person is:

```
<sur:user name="user ID" />
```

The attribute “name” is mandatory and can contain context variables. The number of multi-value context variables is restricted to one variable per `<sur:user .../>` element. See also section 3.1.3 on page 21.

Since the returned value is identical to the value of the “name” attribute, the only advantage of using this query element, is to validate this attribute’s value. The query is only offered with this plug-in to be consistent with the other staff resolution plug-ins delivered with WPC. For performance reasons it is recommended to use the `<sur:userID .../>` query instead.

### 3.3.2 Users Of Group Query

The format to lookup for the user IDs of group members is:

```
<sur:usersOfGroup groupName="group name" />
```

The attribute “groupName” is mandatory and can contain context variables. The number of multi-value context variables is restricted to one variable per `<sur:usersOfGroup .../>` element. See also section 3.1.3 on page 21.

### 3.3.3 Search Query

The User Registry also provides the means to search for a person or group based on a search pattern string. Refer to the User Registry documentation for more information regarding the format of this string.

The format to search for a person or group is defined as in this example:

```
<sur:search type="user"|"group"
           pattern="search pattern" />
```

The attributes “type” and “pattern” are mandatory. The value of attribute “pattern” can contain context variables. The number of multi-value context variables is restricted to one variable per `<sur:search .../>` element. See also section 3.1.3 on page 21.

There is no way to specify whether subgroups are evaluated recursively or not. The default behavior depends on the User Registry implementation configured for the WebSphere Application Server security component.

### 3.3.4 Samples

This example can be used to assign an activity to the members of a group defined in the User Registry:

```
<sur:staffQueries>
  <sur:usersOfGroup groupName="Administrators"/>
</sur:staffQueries>
```

A more complex example makes use of both User Registry specific staff query elements:

```
<sur:staffQueries threshold="30">
  <sur:usersOfGroup groupName="Administrators"/>
  <sur:search type="user" name="Mi*"/>
</sur:staffQueries>
```

It will resolve the user IDs of all members of the “Administrators” group, and of all users whose name begins with “Mi”.

## 3.4 LDAP Staff Resolution Plug-in

The LDAP staff resolution plug-in allows you to utilize organizational information that is available via LDAP directories. It provides functionality to evaluate all members of an LDAP group or to retrieve specific attributes of an LDAP object such as a manager attribute of a person object.

LDAP staff queries are not necessarily restricted to directories that are known to the WebSphere Application Server and that are used for the WebSphere Application Server authentication. However, the user IDs managed in the LDAP directory must match the ones (caller principal name) used by the WebSphere Application Server.

In addition to the three special queries described in section 3.1.2 on page 20, this plug-in also supports the following query elements:

- `<sldap:user .../>`
- `<sldap:usersOfGroup .../>`
- `<sldap:search .../>`
- `<sldap:intermediateResult .../>`

The formal definition of the supported staff queries as XML schema can be found in section A.3.3 on page 40. The XML namespace used by the deployment elements of this plug-in is defined as:

```
xmlns:sldap=
"http://www.ibm.com/schemas/workflow/wswf/plugins/staff/ldap"
```

### 3.4.1 User Query

To locate people based on their distinguished name, the following format is used:

```
<sldap:user dn="distinguished name of the user"
           attribute="attribute to evaluate"
           objectclass="LDAP objectclass of the queried object" />
```

- **dn** is a mandatory attribute that specifies the distinguished name of the LDAP object to be retrieved. The value of attribute “dn” can contain context variables. The number of multi-value context variables is restricted to one variable per `<sldap:user . . . />` element. See also section 3.1.3 on page 21.
- **attribute** is a mandatory attribute that specifies which attribute of the LDAP object contains the value(s) to be returned.
- **objectclass** is a mandatory attribute and allows you to specify the LDAP object class of the object, to which the dn points. See also “Attribute evaluation specifications” below.

### 3.4.2 Users Of Group Query

To locate members of a group based on the group’s distinguished name, the following format is used:

```
<sldap:usersOfGroup groupDN="distinguished name of the group"
                   recursive="yes" | "no">
  attribute evaluation specification*
</sldap:usersOfGroup>
```

- **groupDN** is a mandatory attribute that specifies the distinguished name of the LDAP object to be retrieved, and can contain context variables. The number of multi-value context variables is restricted to one variable per `<sldap:usersOfGroup . . . />` element. See also section 3.1.3 on page 21.
- **recursive** is an optional attribute that specifies whether the retrieved LDAP object will be processed recursive or not, for example, whether subgroups of a group will be processed or ignored. Its value defaults to “yes”.
- The **attribute evaluation specification** is mandatory and allows you to define for example, either which attribute contains the userID to be returned, or the group members whose userIDs will be returned. The number of attributes specified is not limited. See also below.

### 3.4.3 Search Query

To locate people or groups based on an LDAP search, the following format is used:

```
<sldap:search baseDN="search root"
             filter="valid LDAP filter"
             searchScope="subtreeScope" |
                       "onelevelScope" |
```

```
                "objectScope"  
                recursive="yes" | "no">  
    attribute evaluation specification *  
</sldap:search>
```

- **baseDN** is an optional attribute that specifies the LDAP sub-tree, on which the search operation is applied. Tree parent nodes or siblings of the baseDN node will not be taken into consideration. If omitted, the search is executed using the baseDN specified in the plug-in configuration described in section 4.3 on page 34. The value of attribute “baseDN” can contain context variables. The number of multi-value context variables is restricted to one variable per `<sldap:search ... />` element. See also section 3.1.3 on page 21.
- **filter** is a mandatory attribute that allows you to specify a valid LDAP search filter that is applied to all objects below the baseDN, in order to restrict the set of LDAP objects returned. The value of the attribute “filter” can contain context variables. The number of multi-value context variables is restricted to one variable per `<sldap:search ... />` element. See also section 3.1.3 on page 21.
- **searchScope** is an optional attribute that allows you to specify how deep to search below the baseDN:
  - **objectScope** only applies the filter to the object specified by the baseDN.
  - **onelevelScope** applies the filter to all objects at the first level under the baseDN node.
  - **subtreeScope** applies the filter to the entire sub-tree of the baseDN node.

If omitted, the search is executed using the search scope specified in the plug-in configuration described in section 4.3 on page 34.

The search scope allows you to define how many levels of the LDAP tree beneath the baseDN will be evaluated, which is orthogonal to the recursion over links (dn’s). So both kinds of nesting can be exploited in parallel.

- **recursive** is an optional attribute that specifies whether the LDAP objects found will be processed recursively or not, for example, whether subgroups as members of a group will be processed or ignored. Its value defaults to “yes”.
- The **attribute evaluation specification** is mandatory and allows you to define for example, which attribute contains the user ID to be returned, or the group members whose user IDs will be returned. The number of attributes specified is not limited.

### 3.4.4 Attribute Evaluation Specifications

Whereas the “user”, “usersOfGroup” and “search” elements define which LDAP objects are evaluated, the attribute evaluation specification defines how to retrieve the query result from these objects. For each LDAP object class that is to be evaluated, an attribute specification must be defined, which contains the (intermediate) result of interest. The attribute can contain the user ID of the queried person, the dn’s of the group members to be evaluated implicitly or some arbitrary string(s) in case of the intermediate results described in section 3.4.6 on page 31.

With the attribute specification, a relationship is established between the attribute name and the LDAP object class. The attribute name and object class are always used as a pair. LDAP objects are evaluated by their class. If an attribute is specified for its class, then the object will be evaluated. If no attribute is defined for its class, the object will be ignored.

The plug-in supports only those LDAP attributes that have one or more string values. Attributes that contain, for example, a JPEG image of the user, are not supported and an exception will be thrown.

Many LDAP objects have more than one object class. The attribute-objectclass pair specification that is found first, will be used. Thus the order, in which the attributes are declared, is important.

The attribute element is defined as:

```
<slldap:attribute name="attribute name"
                 objectclass="LDAP object class"
                 usage="simple" | "recursive" />
```

- **name** is a mandatory attribute that specifies the name of the attribute to evaluate.
- **objectclass** is a mandatory attribute that specifies the LDAP object class for this attribute. For each LDAP object class that is to be evaluated, an attribute specification must be defined.
- **usage** is a mandatory attribute that allows you to specify how the content of this attribute has to be evaluated:
  - **simple** states that the attribute content will be added to the (intermediate) query result set.
  - **recursive** states that this attribute contains the dn values of objects that can be evaluated.
 

They will be evaluated implicitly if the query is recursive. The number of subsequent recursions is not limited for recursive queries.

If the query is not recursive, only one level of indirection will be processed. If objects with recursive attributes are found after the first indirection, they will be ignored.

If the dn values stored in the attribute are to be evaluated, the plug-in resolves the corresponding objects and evaluates for them again the attribute corresponding to the LDAP object class of the retrieved object(s).

It makes sense to define more than one attribute per query if the query could return LDAP objects of more than one object class or if the query has to be evaluated recursively and during recursion, objects of different classes are to be evaluated.

This leads to the following behavior:

1. Recursive queries:
  - (a) Users of group queries:
    - If the object's target attribute is simple (user), its content will be added to the (intermediate) result set.

- If the object's target attribute is recursive (group), its content must be a set of dn values pointing to other LDAP objects that will be resolved and treated like 1.a (recursion).

(b) Search queries:

A search query returns a set of objects, each of them will be processed individually using the algorithm described in 1.a.

2. Non recursive queries:

(a) Users of group queries:

- If the object's target attribute is simple (user), its content will be added to the (intermediate) result set.
- If the object's target attribute is recursive (group), its content must be a set of dn values pointing to other LDAP objects that will be resolved and treated as follows:
  - If the object's target attribute is simple (user), its content will be added to the (intermediate) result set.
  - If the object's target attribute is recursive (group), it will be ignored (no recursion).

(b) Search queries:

A search query returns a set of objects, each of them will be processed individually using the algorithm described in 2.a.

Thus recursive queries resolve users, users of a group and users of subgroups, regardless of how deep they are nested. Non-recursive queries resolve users, users of a group, but no users of subgroups.

### 3.4.5 Samples for “user”, “usersOfGroup” and “search” Queries

The following examples show how user, usersOfGroup and search queries are processed:

**Simple user retrieval:**

```
<sldap:staffQueries>
  <sldap:user dn="cn=Mary Smith, ou=mydivision, o=acme, c=us"
    objectclass="person"
    attribute="sn"/>
</sldap:staffQueries>
```

Query evaluation sequence:

1. Retrieve the LDAP object (Mary Smith) whose dn value has been specified.
2. The entry in the “sn” attribute is read and added to the result set. It is assumed that the sn value is identical to the WebSphere Application Server authenticated user ID.

**Simple users of group retrieval:**

```

<sldap:staffQueries>
  <sldap:usersOfGroup groupDN="cn=mygroup, o=acme, c=us"
    recursive="no">
    <sldap:attribute name="sn"
      objectclass="person"
      usage="simple"/>
    <sldap:attribute name="member"
      objectclass="groupOfNames"
      usage="recursive"/>
  </sldap:usersOfGroup>
</sldap:staffQueries>
</wf:staff>

```

Query evaluation sequence:

1. Retrieve the object (mygroup) whose dn value has been specified.
2. The entries in the “member” attribute are read and resolved.
3. The resolved member objects are evaluated:
  - Objects of type “person” are resolved and the contents of their non-recursive attribute “sn” are added to the result set. It is assumed that the sn value is identical to the WebSphere Application Server authenticated user ID.
  - Objects of type “groupOfNames” (subgroups) are ignored since the query is not recursive.

#### **Recursive users of group retrieval:**

```

<sldap:staffQueries>
  <sldap:usersOfGroup groupDN="cn=mygroup, o=acme, c=us"
    recursive="yes">
    <sldap:attribute name="sn"
      objectclass="person"
      usage="simple"/>
    <sldap:attribute name="member"
      objectclass="groupOfNames"
      usage="recursive"/>
  </sldap:usersOfGroup>
</sldap:staffQueries>

```

Query evaluation sequence:

1. Retrieve the object (mygroup) whose dn value has been specified.
2. The entries in the “member” attribute are read and resolved.
3. The resolved member objects are evaluated:
  - Objects of type “person” are resolved and the contents of their non-recursive attribute “sn” are added to the result set. It is assumed that the sn value is identical to the WebSphere Application Server authenticated user ID.

- Objects of type “groupOfNames” (subgroups) are also resolved, since the query is recursive. The subgroup members are treated as in step 2. This algorithm repeats recursively until all newly resolved subgroups contain no more subgroups.

**Simple search:**

```
<sldap:staffQueries>
  <sldap:search baseDN="ou=mydivision, o=acme, c=us"
    filter="cn=* "
    searchScope="onelevelScope"
    recursive="no">
    <sldap:attribute name="sn"
      objectclass="person"
      usage="simple"/>
    <sldap:attribute name="member"
      objectclass="groupOfNames"
      usage="recursive"/>
  </sldap:search>
</sldap:staffQueries>
```

**Query evaluation sequence:**

1. Retrieve the LDAP objects that are below the baseDN, and to which the filter applies. In this example, these are all objects at the first tree level under the organizational unit “mydivision”.
2. All objects found are resolved and evaluated.
  - Objects of type “person” are resolved and the contents of their non-recursive attribute “sn” are added to the result set. It is assumed that the sn value is identical to the WebSphere Application Server authenticated user ID.
  - Objects of type “groupOfNames” are resolved and evaluated, and the objects whose dn value is found in the member attribute (group members) are resolved and evaluated as follows:
    - Objects of type “person” are resolved and the contents of their non-recursive attribute “sn” are added to the result set.
    - Objects of type “groupOfNames” are subgroups and are therefore ignored.

**Recursive search:**

```
<sldap:staffQueries threshold="30">
  <sldap:search baseDN="ou=mydivision, o=acme, c=us"
    filter="cn=* "
    searchScope="onelevelScope"
    recursive="yes">
    <sldap:attribute name="sn"
      objectclass="person"
      usage="simple"/>
    <sldap:attribute name="member"
      objectclass="groupOfNames"
      usage="recursive"/>
  </sldap:search>
</sldap:staffQueries>
```

Query evaluation sequence:

1. Retrieve the LDAP objects that are below the baseDN, and to which the filter applies. In this example, these are all objects at the first tree level under the organizational unit “mydivision”.
2. All found objects are resolved and evaluated:
  - Objects of type “person” are resolved and the contents of their non-recursive attribute “sn” are added to the result set. It is assumed that the sn value is identical to the WebSphere Application Server authenticated user ID.
  - Objects of type “groupOfNames” (groups/subgroups) are resolved and evaluated, and the objects whose dn value is found in the member attribute (group members) are resolved and evaluated as in step 2. This algorithm repeats recursively until all newly resolved subgroups have no more own subgroups.

### 3.4.6 Queries of type “intermediateResult”

For some queries, the desired result cannot be accomplished with a single query statement. To get the user ID of the manager of the process starter, for example, first the manager attribute (its dn) has to be resolved and only in a second query, can the manager’s user ID be resolved.

The query type “intermediateResult” allows such queries to be performed. It allows you to encapsulate a single query of type `<sldap:user .../>`, `<sldap:usersOfGroup .../>` or `<sldap:search .../>`, and stores the result in a buffer whose content can be used like a context variable in subsequent queries. The result of an intermediateResult query element is not stored in the global query result set.

The result of the query inside an “intermediateResult” element must be a set of strings. No other kind of result objects can be used as a context variable.

The intermediateResult element is defined as:

```
<sldap:intermediateResult name="variable name"
                          threshold="positive nonzero integer number"
                          query of type user, usersOfGroup or search
</sldap:intermediateResult>
```

- **name** is a mandatory attribute that specifies the name of the context variable, in which the intermediate result will be stored. This variable is only valid inside the `<sldap:staffQueries>` element where it is declared and only below its declaration. The name of the context variable must not contain any “%” signs. In case of name collisions with context variables resolved by the engine, the intermediate result takes precedence over any engine variables with the same name. See also section 3.1.3 on page 21 on how to use context variables.
- **threshold** is an optional attribute that allows you to limit the number of items in the context variable string set. If the attribute is not specified, the value of the threshold attribute of the `<sldap:staffQueries>` element is used (it defaults to 2,147,483,647 if not defined there either).

- The actual **query** must be of the type `<sldap:user .../>`, `<sldap:usersOfGroup .../>` or `<sldap:search .../>` and it must return a set of strings. The number of specified queries is limited to exactly one.

This example outlines how to retrieve the manager of the process starter:

```
<sldap:staffQueries>
  <sldap:intermediateResult name="manager">
    <sldap:search baseDN="ou=mydivision, o=acme, c=us"
      filter="uid=%wf:process.starter%"
      searchScope="onelevelScope"
      recursive="no">
      <sldap:attribute name="manager"
        objectclass="inetOrgPerson"
        usage="simple"/>
    </sldap:search>
  </sldap:intermediateResult>
  <sldap:user dn="%manager%"
    attribute="uid"
    objectclass="inetOrgPerson"/>
</sldap:staffQueries>
```

Query evaluation sequence:

- The dn value of the manager of the process starter is retrieved. It is assumed that the “manager” attribute of every user contains the dn value of his or her manager and that the “uid” attribute contains the WebSphere Application Server authenticated user ID.
- The query result is stored in the context variable “manager”. It contains the manager’s dn value.
- The actual query result are retrieved by the user query, it contains the manager’s user ID. The intermediate result “manager” is used like any normal context variable. It is assumed that the “uid” attribute contains the WebSphere Application Server authenticated user ID.

## CHAPTER 4

---

### Staff Plug-in Provider Parameter Reference

---

This chapter describes the parameters needed for the configuration of staff plug-in providers in the WebSphere Application Server Administration Console.

WebSphere Application Server Enterprise comes with three default staff plug-in provider configurations, which are set up during the installation process. One provider will be configured for the System staff resolution plug-in, which is not intended for production environments, another one that will run out of the box is set up for the User Registry plug-in, and a third one that will have to be customized in most of the cases, in order to be able to connect to the right LDAP server, will be configured for the LDAP plug-in.

Thus, you must customize the default plug-ins or create further staff plug-in provider configurations in order to be able to connect to different staff repositories or to use customized XSLT staff verb mapping files.

For a better understanding of the staff resolution plug-ins and the staff plug-in provider, refer to [3] and [4] in section B on page 45.

This chapter describes the configuration parameters supported by the various staff resolution plug-ins.

#### **4.1 System Staff Resolution Plug-in**

This plug-in requires no configuration parameters.

#### **4.2 User Registry Staff Resolution Plug-in**

The User Registry plug-in does not require any configuration parameters, as it is based on the WebSphere Application Server context and configuration. Nevertheless, it makes use of the information about the concrete class implementing the UserRegistry interface, and the User Registry configuration parameter `WAS_UsedisplayName` that can

be set for custom registries, to be able to always return the User Registry parameter corresponding to the caller principal name as query result.

### 4.3 LDAP Staff Resolution Plug-in

The LDAP staff resolution plug-in requires several deployment parameters that are managed the Staff Support Service administration user interface. The plug-in connects to the LDAP server using JNDI, most configuration parameters are needed to set up this connection.

The configuration parameters are:

- **AuthenticationAlias** is an optional parameter. When not set, anonymous logon to the LDAP server is used. This parameter represents the authentication alias used to connect to LDAP. The alias must have been previously defined in the WebSphere Application Server Administration Console under Security → JAAS Configuration → J2C Authentication Data. Sample: “mycomputer/My LDAP Alias”.
- **AuthenticationType** is an optional parameter. By default, the logon will be anonymous if the **AuthenticationAlias** is not set, otherwise it will use simple authentication.
- **BaseDN** is a mandatory parameter that defines the default base DN for all LDAP search operations, for example, “o=acme, c=us”.
- **ContextFactory** is a mandatory parameter that sets the JNDI context factory Java™ class, for example, “com.sun.jndi.ldap.LdapCtxFactory”.
- **ProviderURL** is a mandatory parameter that sets the URL pointing to the LDAP/JNDI directory server and port. The usual JNDI syntax applies here, for example, “ldap://localhost:389”.
- **SearchScope** is a mandatory parameter that defines the default search scope for LDAP search operations. Only the values “objectScope”, “onelevelScope” and “subtreeScope” are allowed.
- **additionalParameterName1-5** together with **additionalParameterValue1-5** allow you to set arbitrary additional JNDI properties for the connection to the LDAP server. With these optional configuration parameters, up to five JNDI parameters can be specified as name-value pairs.

# APPENDIX A

---

## XML schemas

---

### A.1 Staff Query Verb Set

The staff query verb set is used by WebSphere Studio Application Developer Integration Edition to define staff resolution queries.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/verbset"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:vs=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/verbset"
  xmlns=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/verbset"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified">

  <xsd:element name="VerbSet">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Root element
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
        <xsd:element ref="DefineVerb" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="Verb">
      <xsd:selector xpath="./vs:DefineVerb"></xsd:selector>
      <xsd:field xpath="@name"></xsd:field>
    </xsd:key>
  </xsd:element>
```

```
<xsd:element name="DefineVerb">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
      <xsd:element name="Mandatory">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="Parameter"
              minOccurs="0"
              maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Optional">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="Parameter"
              minOccurs="0"
              maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/></xsd:minLength>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:key name="Parameter">
    <xsd:selector xpath=
      ".//vs:Mandatory/vs:Parameter | .//vs:Optional/vs:Parameter"/>
    <xsd:field xpath="vs:Name"/>
  </xsd:key>
</xsd:element>

<xsd:element name="Parameter">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:NCName"/>
      <xsd:element name="Type" type="dataType"/>
      <xsd:element
        name="Hint"
        type="xsd:string"
        minOccurs="0"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:simpleType name="dataType">
  <xsd:restriction base="xsd:Name">
    <xsd:enumeration value="xsd:boolean"/>
    <xsd:enumeration value="xsd:date"/>
    <xsd:enumeration value="xsd:dateTime"/>
    <xsd:enumeration value="xsd:decimal"/>
  </xsd:restriction>
</xsd:simpleType>
```

```

    <xsd:enumeration value="xsd:double" />
    <xsd:enumeration value="xsd:duration" />
    <xsd:enumeration value="xsd:float" />
    <xsd:enumeration value="xsd:gDay" />
    <xsd:enumeration value="xsd:gMonth" />
    <xsd:enumeration value="xsd:gMonthDay" />
    <xsd:enumeration value="xsd:gYear" />
    <xsd:enumeration value="xsd:gYearMonth" />
    <xsd:enumeration value="xsd:integer" />
    <xsd:enumeration value="xsd:string" />
    <xsd:enumeration value="xsd:time" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

## A.2 Parameterized Verb

The parameterized verb is the input document for the XSLT verb mapping.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ibm.com/schemas/workflow/wswf/plugins/staff"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns="http://www.ibm.com/schemas/workflow/wswf/plugins/staff">

  <xsd:element name="verb">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="id" type="xsd:string"/>
        <xsd:element name="parameter" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="typeUnion">
                <xsd:attribute name="id" type="xsd:string"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="typeUnion">
    <xsd:union memberTypes="xsd:string xsd:boolean"/>
  </xsd:simpleType>

</xsd:schema>

```

## A.3 Staff Resolution Plug-ins

The staff resolution plug-ins are responsible for the deployment and execution of staff resolution queries.

### A.3.1 System Staff Resolution Plug-in

The System staff resolution plug-in allows you to define simple static staff assignments. It is not intended for production and only supports the staff query types “everybody”, “nobody” and “userID”:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/system"
  xmlns="http://www.ibm.com/schemas/workflow/wswf/plugins/staff/system"
  version="1.0"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified">

  <xsd:element name="staffQueries">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="everybody" type="everybody"/>
        <xsd:element name="nobody" type="nobody"/>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="userID" type="userID"/>
        </xsd:choice>
      </xsd:choice>
      <xsd:attribute
        name="threshold"
        type="xsd:positiveInteger"
        default="2147483647"
        use="optional">
        <xsd:annotation>
          <xsd:documentation xml:lang="en-US">
            Limits the number of returned user ID values.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="everybody"/>

  <xsd:complexType name="nobody"/>

  <xsd:complexType name="userID">
    <xsd:attribute name="name" type="xsd:string" use="required">
      <xsd:annotation>
        <xsd:documentation xml:lang="en-US">
          Must be a (or resolve to one or more) valid WebSphere user ID.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
</xsd:schema>
```

### A.3.2 User Registry Staff Resolution Plug-in

The User Registry staff resolution plug-in allows you to define staff assignments using the WebSphere User Registry as the staff repository. It supports the staff query types “everybody”, “nobody”, “search”, “user”, “userID” and “usersOfGroup”:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/userregistry"
  xmlns=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/userregistry"
  version="1.0"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified">

  <xsd:element name="staffQueries">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="everybody" type="everybody"/>
        <xsd:element name="nobody" type="nobody"/>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="search" type="search"/>
          <xsd:element name="user" type="user"/>
          <xsd:element name="userID" type="userID"/>
          <xsd:element name="usersOfGroup" type="usersOfGroup"/>
        </xsd:choice>
      </xsd:choice>
      <xsd:attribute
        name="threshold"
        type="xsd:positiveInteger"
        default="2147483647"
        use="optional">
        <xsd:annotation>
          <xsd:documentation xml:lang="en-US">
            Limits the number of returned user ID values.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="everybody"/>

  <xsd:complexType name="nobody"/>

  <xsd:complexType name="search">
    <xsd:attribute name="type" use="required" type="type"/>
    <xsd:attribute name="pattern" use="required" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation xml:lang="en-US">
          UserRegistry search pattern.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
```

```
<xsd:complexType name="user">
  <xsd:attribute name="name" use="required" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Name of the user.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="userID">
  <xsd:attribute name="name" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Must be a (or resolve to one or more) valid WebSphere user ID.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="usersOfGroup">
  <xsd:attribute name="groupName" use="required" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Name of the group.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:simpleType name="type">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="user" />
    <xsd:enumeration value="group" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

### A.3.3 LDAP Staff Resolution Plug-in

The LDAP staff resolution plug-in allows you to define staff assignments based on LDAP staff repositories. It supports the staff query types “everybody”, “nobody”, “search”, “user”, “userID”, “usersOfGroup” and “intermediateResult”:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace=
    "http://www.ibm.com/schemas/workflow/wswf/plugins/staff/ldap"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.ibm.com/schemas/workflow/wswf/plugins/staff/ldap"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.0">

  <xsd:element name="staffQueries">
```

```

<xsd:complexType>
  <xsd:choice>
    <xsd:element name="everybody" type="everybody"/>
    <xsd:element name="nobody" type="nobody"/>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element name="intermediateResult"
        type="intermediateResult"/>
      <xsd:element name="search" type="search"/>
      <xsd:element name="user" type="user"/>
      <xsd:element name="userID" type="userID"/>
      <xsd:element name="usersOfGroup" type="usersOfGroup"/>
    </xsd:choice>
  </xsd:choice>
  <xsd:attribute name="threshold"
    type="xsd:positiveInteger"
    use="optional"
    default="2147483647">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Limits the number of returned user ID values.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
</xsd:element>

<xsd:complexType name="everybody"/>

<xsd:complexType name="intermediateResult">
  <xsd:choice>
    <xsd:element name="search" type="search"/>
    <xsd:element name="user" type="user"/>
    <xsd:element name="usersOfGroup" type="usersOfGroup"/>
  </xsd:choice>
  <xsd:attribute name="name" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Name of the intermediate result.
        Can be referenced in subsequent queries as variable
        like e.g. filter="&quot;cn=%var%&quot;.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="threshold" type="xsd:string" use="optional">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Limits the number of returned intermediateResult values.
        When not set, the value of the threshold attribute
        from the element &lt;staffQueries&gt; (or its default)
        will be used.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="nobody"/>

<xsd:complexType name="search">

```

```
<xsd:sequence>
  <xsd:element name="attribute"
    type="attribute"
    maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="baseDN" type="xsd:string" use="optional">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
      Startpoint distinguished name (dn) for the search.
    </xsd:documentation>
  </xsd:annotation>
</xsd:attribute>
<xsd:attribute name="filter" type="xsd:string" use="required">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
      LDAP search filter.
    </xsd:documentation>
  </xsd:annotation>
</xsd:attribute>
<xsd:attribute name="recursive"
  type="recursive"
  use="optional"
  default="yes">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
      Defines whether the result shall be evaluated recursive or not.
    </xsd:documentation>
  </xsd:annotation>
</xsd:attribute>
<xsd:attribute name="searchScope" type="searchScope" use="optional"/>
</xsd:complexType>

<xsd:complexType name="user">
  <xsd:attribute name="dn" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Distinguished name (dn) of the user.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="attribute" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Attribute containing the user ID of this user.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="objectclass" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        LDAP objectclass of the user object.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="userID">
  <xsd:attribute name="name" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
```

```
        Must be a (or resolve to one or more) valid WebSphere user ID.
    </xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>

<xsd:complexType name="usersOfGroup">
  <xsd:sequence>
    <xsd:element name="attribute"
      type="attribute"
      maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="groupDN" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Distinguished name (dn) of the group.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="recursive"
    type="recursive"
    use="optional"
    default="yes">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Defines whether the result shall be evaluated recursive or not.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>

<xsd:complexType name="attribute">
  <xsd:attribute name="name" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Name of the LDAP attribute to evaluate.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="objectclass" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
        Valid LDAP objectclass.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="usage" type="usage" use="required" />
</xsd:complexType>

<xsd:simpleType name="recursive">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="yes" />
    <xsd:enumeration value="no" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="searchScope">
  <xsd:restriction base="xsd:string">
```

```
        <xsd:enumeration value="subtreeScope"/>
        <xsd:enumeration value="onelevelScope"/>
        <xsd:enumeration value="objectScope"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="usage">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="recursive"/>
        <xsd:enumeration value="simple"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

## APPENDIX B

---

### References

---

1. Matthias Kloppmann, Gerhard Pfau. WebSphere Application Server Enterprise Process Choreographer - Concepts and Architecture. December 2002.
2. Günther Jornitz, Matthias Kloppmann, Gerhard Pfau, Stefan Rüttinger. WebSphere Platform Programming Model - Business Process Choreography. To be published.
3. Kurt Lind, Eric Van Norman. WebSphere Application Server Enterprise Process Choreographer - Staff Resolution Architecture. March 2003.
4. Kurt Lind, Eric Van Norman. WebSphere Platform Programming Model - Staff Support Service. To be published.
5. WebSphere Studio Application Developer Integration Edition 5.0 online help



## APPENDIX C

---

### Trademark attributions and copyrights

---

© Copyright IBM Corporation 2003. All rights reserved.

IBM and WebSphere are trademarks of the IBM Corporation in the United States, other countries, or both.

#### **Special attributions**

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.