

Advanced Snapin For ConsoleOne

Documentation

www.novell.com

May 2003



Novell[®]

Disclaimer Novell, Inc. makes no representations or warranties with respect to the contents or use of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication, and to make changes to its content, at any time.

Further, Novell, Inc. makes no representations or warranties with respect to any Novell software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Trademarks Novell and NetWare are registered trademarks of Novell, Inc. in the United States and other countries.

The Novell Network Symbol is a trademark of Novell, Inc.

* All third-party trademarks are property of their respective owner.

Copyright © 2000 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of Novell Consulting.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U. S. A.

Prepared By Volker Scheuber

Documentation

May 2003

3rd Party Acknowledgements This product includes software developed by:

John Clark (<http://www.jclark.com/>)

Rob Rawson

Document Revision History

Date	Build	Author	Description
2000-11-28	20001128-0	Volker Scheuber	First draft
2000-12-04	20001204-0	Volker Scheuber	Updated DTD documentation and smples to reflect changes in adv.dtd due to enhancements in the synchronization functionality. Added chapter about synchronization.
2000-12-05	20001209-0	Volker Scheuber	Updated point 12 in chapter "Configuration".
2000-12-19	20001221-0	Volker Scheuber	Added documentation for new GUI element <dn-panel> Completed DTD documentation for text areas
2001-02-16	20010216	Volker Scheuber	Added chapter "Supported Syntax Mapping"
2001-02-18	20010216	Volker Scheuber	Completed DTD documentation for the new supported syntaxes.
2001-02-22	20010222	Volker Scheuber	Added chapter "Troubleshooting Advanced Snapins" with screenshots. Added two more examples with XML code and screenshots.
2001-03-04	20010304	Volker Scheuber	Completed DTD documentation for combo boxes. Added combo box examples with XML code and screenshots.
2001-06-25	20010625	Volker Scheuber	Completed DTD documentation for pictures. Added picture examples with XML code and screenshots.
2001-07-11	20010711	Volker Scheuber	Updated chapter "Manual Installation".
2001-09-23	20010923	Volker Scheuber	Completed DTD documentation for "absolute" layout. Added example for "absolute" layout ("mySecondSnapin"). Added example for "xml-panel". Updated syntax mapping table for SYN_TIME support.
2001-11-15	20011116	Volker Scheuber	Updated documentation to conform to the new CI.
2001-11-17	20011116	Volker Scheuber	Completed DTD documentation to contain new attributes for dn-panels: show-name-only and start-context. Updated example for xml-panel. Added example for creator snapins. Added example for icon snapins.
2001-12-05	20020226	Volker Scheuber	Minor corrections to DTD documentation. Updated DTD documentation to contain new attribute: action. Added chapter "Actions / Functions".
2002-01-06	20020226	Volker Scheuber	Updated function index
2002-02-27	20020226	Volker Scheuber	Updated function index Updated DTD documentation to contain new elements: before-creation-action and after-creation-action. Added examples for actions and plug-ins.
2002-05-21	2002-05-21	Volker Scheuber	Corrected errors in function index
2002-12-30	0.9.13	Volker Scheuber	Added DTD documentation for - <button> Updated DTD documentation for

			- <dn-panel>
2003-05-12	1.0.0	Volker Scheuber	Completed chapter "Plug-Ins" with a complete "Plug-In Index".

Contents

Introduction	10
Overview	10
What Is "Advanced Snapin"?	10
What Problems Solves "Advanced Snapin"?	10
What Are the Key Features of "Advanced Snapin"?	10
Architecture	11
Overview	11
New Object Classes	11
AdvSnapinContainer	11
AdvSnapinClass	11
AdvSnapinType	11
AdvSnapin	11
Installation	13
Prerequisites	13
Using the Installer (Windows platforms only)	13
Manual Installation	13
Configuration	14
Troubleshooting Advanced Snapins	17
Document Type Definition "adv.dtd"	18
Top Level Element	18
adv	18
Snapin Elements	20
page-snapin	20
creator-snapin	21
Input Elements	24
check-box	24
combo-box	25
dn-panel	27
item	29
label	31
picture	31
radio-button	34
text-area	36
text-field	38
xml-panel	41
Special Elements	43

border.....	43
button.....	44
before-creation-action	45
after-creation-action	45
Supported Syntax Mapping	47
Synchronization	49
Actions - Functions	50
Overview	50
Functions.....	50
Syntax.....	50
Function Index	51
Sample Actions	52
Plug-Ins	53
Overview	53
How to Execute Plug-Ins	53
Plug-In Index	53
MyFirstPlugin.....	53
Check Unique Attribute	53
Unique Name Generator.....	54
GUI Spy	55
ID Server CLient	55
ID Server Remote Control	55
Dynamic Combo Box	56
Convert Leaf to Root Most	56
Open Property Page.....	56
amHeritage.....	56
parseMe.....	57
parseMeToo	57
Selection List Control	57
Sample Snap-Ins	58
Screenshot of "ComboBoxes" in Action	67

Introduction

Overview

What Is “Advanced Snapin”?

Advanced Snapin is a piece of software that registers to Novell’s ConsoleOne as a set of snapins. It allows an administrator to define it’s own user interfaces for ConsoleOne using industry standard XML code. Advanced Snapin then interprets this XML code during runtime. Actually, it has part of the functionality that ScheMax had for NWAdmin.

What Problems Solves “Advanced Snapin”?

Advanced Snapin let’s you do this easily.

What Are the Key Features of “Advanced Snapin”?

You may want to check the “readme.txt” that came with this release for further restrictions or enhancements.

- ConsoleOne snapin (user interface) creation within 5 minutes.
- Handles almost all attributes syntaxes.
- Synchronization of “dn” attributes (e.g. “Membership” of class “User” and “Member” of class “Group”).
- Support for multi valued attributes.
- Automatic input element configuration (only have to specify handled attribute’s name and the rest is taken from the attribute definition in the schema).
- Supports creator snapin for existing and user defined object classes.
- Supports pictures.
- Supports a set of commands to dynamically and automatically calculate values of input elements, e.g. automatically calculate the “Full Name” out of the “Given Name” and the “Surname”.
- API to write your own Java Plug-Ins for Advanced Snapin. Plug-ins extend ConsoleOne’s functionality in a different way than snap-ins do: A plug-in might, e.g., call an external Program, launch a wizard or simply display a message box.

Architecture

Overview

Advanced Snapin stores your snapin definitions in eDirectory objects and therefore takes advantage of the directory: As the snapins are stored in the directory it self they can be accessed from everywhere.

New Object Classes

To store the snapin definitions, new object classes are needed.

AdvSnapinContainer

AdvSnapinContainer-objects are some sort of the "root" of all of your Advanced Snapins and do not store any kind of snapin data. Only one AdvSnapinContainer per tree ist supported, which can be placed anywhere in the tree. AdvSnapinContainers can only contain AdvSnapinClass-objects.

AdvSnapinClass

The names of AdvSnapinClass-objects, e.g. "User", specify the class the snapins should manage and are used to organize snapins in the tree so that the Advanced Snapin can quickly retrieve them. No snapin data is stored in the objects itself. AdvSnapinClasses can only contain AdvSnapinType-objects.

AdvSnapinType

The names of AdvSnapinType-objects specify the snapin type, e.g. "Page" (= property page snapins), and are used to organize snapins in the tree so that the Advanced Snapin can quickly retrieve them. No snapin data is stored in the objects itself. AdvSnapinTypes can only contain AdvSnapin-objects.

AdvSnapin

This object class actually stores the snapin definitions in XML format. AdvSnapins can only be placed in AdvSnapinType containers.

Installation

Prerequisites

- ConsoleOne 1.2 or later installed.
NOTE: To take the most advantage of all features of Advanced Snapin, you should have ConsoleOne 1.3.3 or later installed.
- Snapin users must have browse rights to the "AdvSnapinContainerDN" attribute of the root (this attribute is added with an auxiliary class to the root object during the installation process).

Using the Installer (Windows platforms only)

1. Quit ConsoleOne, if running.
2. Run the install file "AdvancedSnapin-yyyymmdd.exe".
3. Follow the instructions on the screen.
4. Start ConsoleOne.

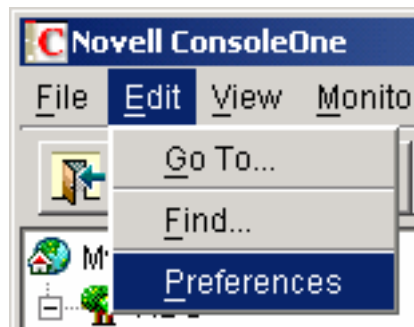
Manual Installation

5. Quit ConsoleOne, if running.
6. Unzip the file "AdvancedSnapin-yyyymmdd.zip" to a temporary directory.
7. Copy the file "AdvSnapin.jar" to the "<ConsoleOne Root>\snapins\Advanced" directory.
8. Copy the files "AdvSnapinLib.jar", "AdvPluginLib.jar", "xp.jar" and "dirxml.jar" to the "<ConsoleOne Root>\lib\Advanced" directory.
9. Copy the file "AdvSnapinRes.jar" to the "<ConsoleOne Root>\resources\Advanced" directory.
10. Start ConsoleOne.

Configuration

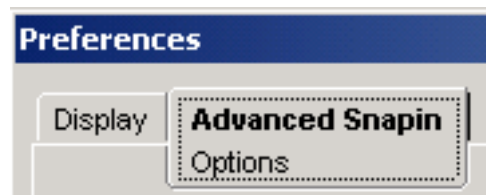
1. Follow the instructions given in chapter "Installation".
2. Select a tree (click with the mouse into an eDirectory tree in the left panel of ConsoleOne) where you want to configure snapins for.
3. Make sure you are logged in to that tree with admin rights to the root.
4. Select "Preferences" from the "Edit" menu.

Configuration - Advanced Snapin Preferences



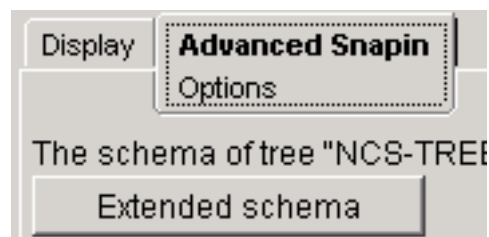
5. Choose the "Options" menu from the "Advanced Snapin" tab.

Configuration - "Advanced Snapin" Tab



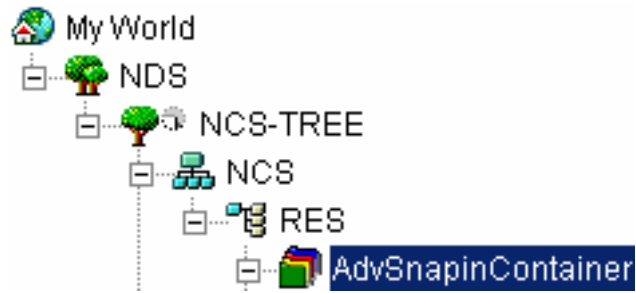
6. Press the button "Extend schema".

Configuration - "Extend Schema"-button



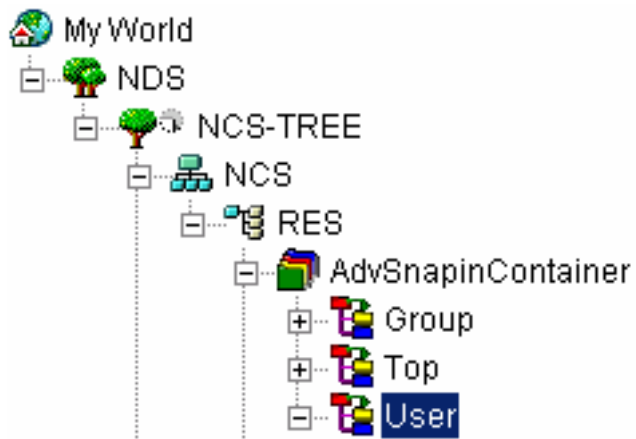
7. Press "OK".
8. Create an "AdvSnapinContainer"-object in the tree. E.g.:
mySnapinContainer.ou.o.myTree

Configuration - "AdvSnapinContainer"-object



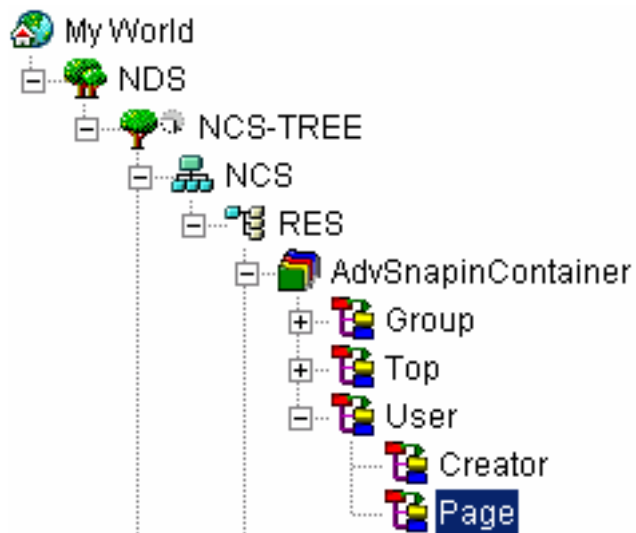
9. Create "AdvSnapinClass" -objects for every class you want to manage. E.g.:
 - User.mySnapinContainer.ou.o.myTree
 - Group.mySnapinContainer.ou.o.myTree

Configuration - "AdvSnapinClass"-objects



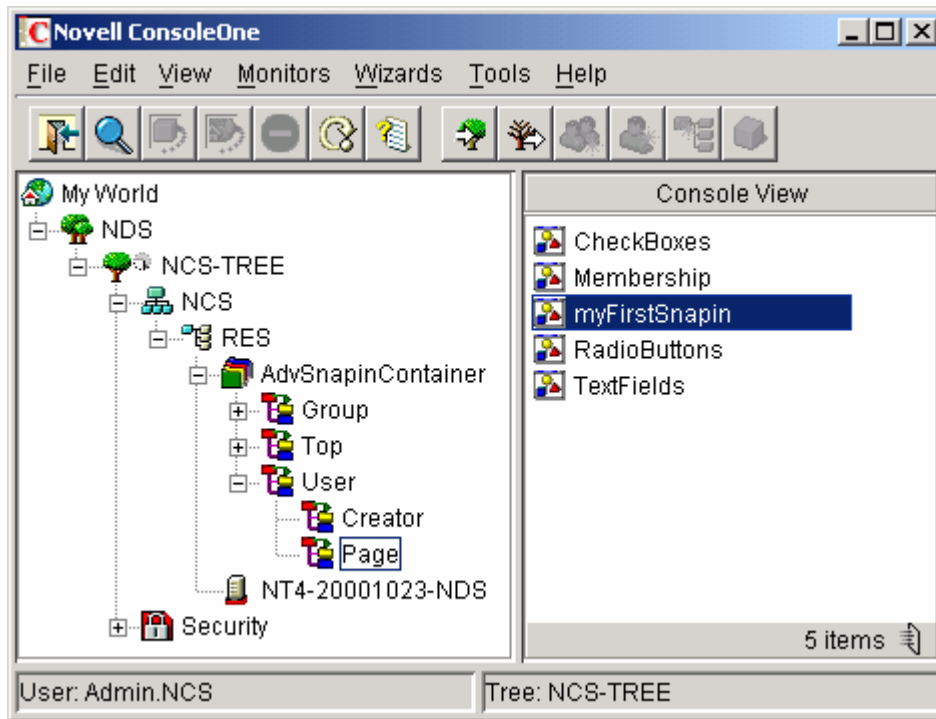
10. Create "AdvSnapinType" -objects for your snapins. E.g.:
 - Page.User.mySnapinContainer.ou.o.myTree
 - Page.Group.mySnapinContainer.ou.o.myTree

Configuration - "AdvSnapinType"-objects



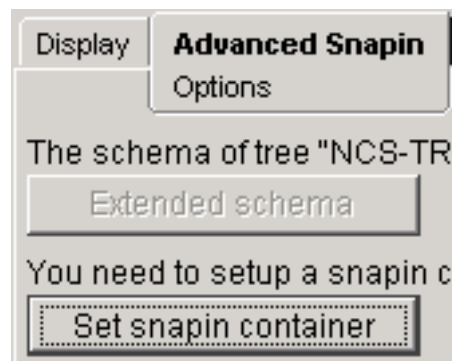
11. Create "AdvSnapin" -objects as leaves into the new tree structure to represent your snapins. E.g.:
 - myUserPageSnapin.Page.User.mySnapinContainer.ou.o.myTree
 - myGroupPageSnapin.Page.Group.mySnapinContainer.ou.o.myTree

Configuration - Possible tree structure for Advanced Snapin



12. Enter your snapin definition into the "XmlData" attribute of your "AdvSnapin" -objects (use the "Other" tab).
 Note: Avoid DOCTYPE tags like:
`<!DOCTYPE adv SYSTEM "E:\My Documents\Projects\Advanced\Snapin\XML\adv.dtd">`
 They will prevent the snapin from loading.
13. Select "Preferences" from the "Edit" menu.
14. Choose the "Options" menu from the "Advanced Snapin" tab.
15. Press the "Set snapin container" button.

Configuration - "Set snapin container"-button

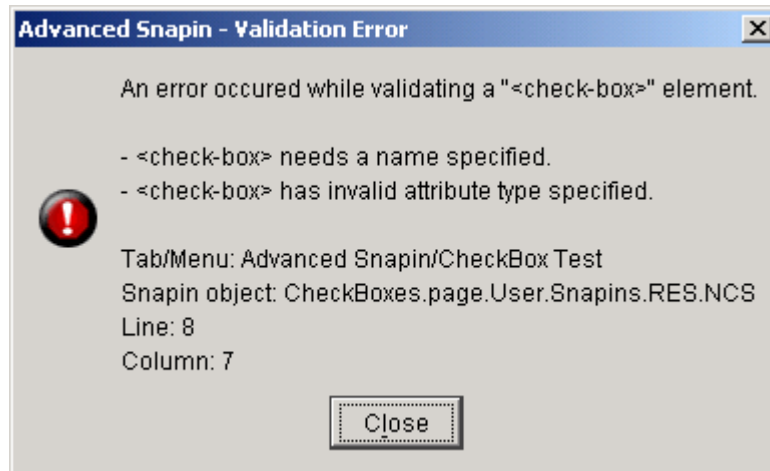


16. Select your new "AdvSnapinContainer" (see 8.)
17. Press OK.
18. Your snapins become active immediately.

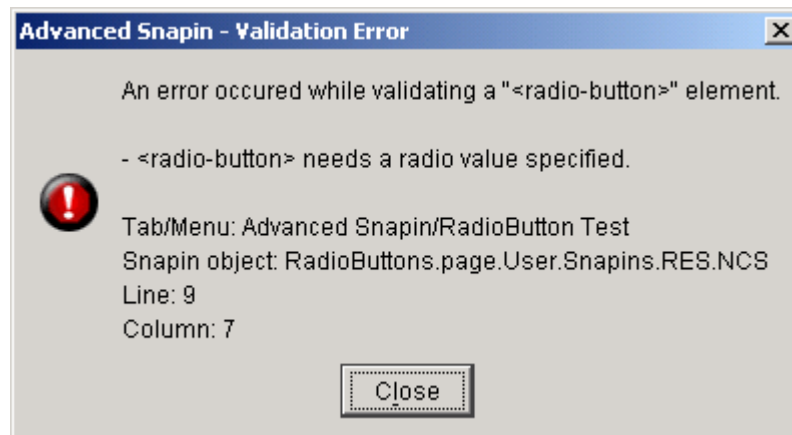
Troubleshooting Advanced Snapins

Advanced Snapin tries to tell you what's wrong if ever possible. It will show you descriptive error messages which tell you exactly in which snapin object the error occurred, on which line and column. See the examples below:

Troubleshooting Advanced Snapins - Sample error message 1



Troubleshooting Advanced Snapins - Sample error message 2



Document Type Definition “adv.dtd”

The ADV document type definition file (adv.dtd) defines the schema of the XML documents that the Advanced Snapin can process. XML documents that do not conform to this schema generate errors.

Top Level Element

All XML documents for the Advanced Snapin must start with a top-level element. This is in all cases <adv> with some attributes.

adv

Specifies the top level element that is used in all documents used as Advanced Snapin definitions.

Description

All XML documents must use adv as the top level element in the document.

Definition

```
<!ELEMENT adv (page-snapin?)>
<!ATTLIST adv
  dtd CDATA #REQUIRED
  version CDATA #REQUIRED
>
```

Attributes of adv

dtd

Specifies the current DTD. It must be set to “adv.dtd”

version

Specifies the current version of Advanced Snapin. It must be set to “1.0”.

Elements of adv

page-snapin

Specifies a property page snapin.

creator-snapin

Specifies a creator panel snapin. Creator panel snapins are called whenever an object of a specific class is being created. A creator snapin must provide fields for at least all mandatory attributes of that object class.

Parent

None.

Example

```
<adv dtd="adv.dtd" version="1.0">
  <creator-snapin
    snapin-name="myFirstCreatorSnapin"
    snapin-description="My first creator snapin"
    layout="absolute">
    <label
      constraints="5,5,210,20"
      name="Label01"
      text="Full Name: "/>
    <text-field
      constraints="5,30,210,20"
      name="TextField101"
      attr-name="Full Name"
      multi-valued="false"/>
  </creator-snapin>
</adv>
```

Snapin Elements

page-snapin

Specifies a property page snapin. This is the type of ConsoleOne snapins used most often.

Description

This tag contains important information about your snapin, such as name, tab name, menu name, description etc. It is also the parent of all input elements.

Definition

```
<!ELEMENT page-snapin (check-box* | combo-box* | label* | radio-button* | text-area*
| text-field*)>
<!ATTLIST page-snapin
  snapin-name CDATA #REQUIRED
  snapin-description CDATA #REQUIRED
  snapin-tab-name CDATA #REQUIRED
  snapin-menu-name CDATA #REQUIRED
  class CDATA #IMPLIED
  scope (creator | page) #IMPLIED
>
```

Attributes of page-snapin

snapin-name

(Invisible) name of your snapin.

snapin-description

(Invisible) description of your snapin.

snapin-tab-name

Specifies the name of the tab where your page snapin will register on. This can be a new, e.g. "my Snapins" or an existing tab, e.g. "General".

snapin-menu-name

Specifies the name of the menu on the tab where you want your snapin to register. Note that this name must be unique on its tab.

layout

Specifies the snapin type. Possible values are: "standard" and "absolute".

Class

Specifies the name of the object class you want to manage.

Scope

Specifies the snapin type. Possible values are: "Page".

Elements of page-snapin

- check-box
- combo-box
- dn-panel
- label
- radio-button
- text-area
- text-field
- xml-panel
- border

Parent

adv

Example

```
<adv dtd="adv.dtd" version="1.0">
  <page-snapin
    snapin-name="Membership"
    snapin-description="Membership"
    snapin-tab-name="Advanced Snapin"
    snapin-menu-name="Membership"
    class="User"
    scope="page">
    <text-field
      name="TextField1"
      label="Member of"
      attr-name="Group Membership"
      attr-type="dn"
      multi-valued="true"
      sync="Member"/>
    </page-snapin>
  </adv>
```

creator-snapin

Specifies a creator panel snapin. This is the type of ConsoleOne snapins that is used to create a new object of a specific object class e.g. User.

Description

This tag contains important information about your snapin, such as name and description etc. It is also the parent of all input elements.

Definition

```
<!ELEMENT creator-snapin (check-box* | combo-box* | label* | radio-button* | text-
area* | text-field* | dn-panel*)>
<!ATTLIST creator-snapin
  snapin-name CDATA #REQUIRED
  snapin-description CDATA #REQUIRED
  class CDATA #IMPLIED
  scope (creator | page) #IMPLIED
  layout (standard | absolute) "standard"
>
```

Attributes of creator-snapin

snapin-name

(Invisible) name of your snapin.

snapin-description

(Invisible) description of your snapin.

layout

Specifies the snapin type. Possible values are: "standard" and "absolute".

Class

Specifies the name of the object class you want to manage.

Scope

Specifies the snapin type. Possible values are: "Page".

Elements of creator-snapin

- check-box
- combo-box
- dn-panel
- label
- radio-button
- text-area
- text-field
- xml-panel
- border
- before-creation-action
- after-creation-action

Parent

adv

Example

```
<adv dtd="adv.dtd" version="1.0">
  <creator-snapin
    snapin-name="myFirstCreatorSnapin"
    snapin-description="My first creator snapin"
    layout="absolute">
    <frame
      constraints="0,0,210,50"
      name="Frame01"
      text="Additional Personal Information"/>
    <label
      constraints="5,5,210,20"
      name="Label01"
      text="Full Name: "/>
    <text-field
      constraints="5,30,210,20"
      name="TextField01"
      attr-name="Full Name"
```

```
        multi-valued="false"/>  
    </creator-snapin>  
</adv>
```

Input Elements

check-box

Specifies a check box input element.

Description

Specifies a check box input element.

Definition

```
<!ELEMENT check-box EMPTY>
<!ATTLIST check-box
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  attr-name CDATA #REQUIRED
  attr-type (string | state | dn | int) "string"
  text CDATA #IMPLIED
  sync-source CDATA #IMPLIED
  sync-target CDATA #IMPLIED
  action CDATA #IMPLIED
>
```

Attributes of check-box

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

text

Set's the text to be displayed to the right of the input element.

sync-source

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the object you are editing (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

sync-target

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the target object, specified by the dn value (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

action

This lets you execute an Advanced Snapin command.

Elements of check-box

None.

Parent

page-snapin

creator-snapin

Example

```
<check-box
  name="CheckBox1 "
  attr-name="Login Disabled"
  attr-type="boolean"
  label="Account: "
  text="disabled"/>
```

combo-box

Specifies a combo box input element.

Description

Specifies a combo box input element.

Definition

```
<!ELEMENT combo-box (item*)>
<!ATTLIST combo-box
  name CDATA #IMPLIED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  required (true | false) "false"
  editable (true | false) "false"
  rows CDATA "-1"
  attr-name CDATA #REQUIRED
  attr-type (string | state | dn | int) "string"
  sync-source CDATA #IMPLIED
  sync-target CDATA #IMPLIED
```

```
> action CDATA #IMPLIED
```

Attributes of <combo-box>

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

rows

Specifies the amount of rows displayed in this combo box.

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

sync-source

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the object you are editing (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more then one attribute name by separating each name with a comma ",".

sync-target

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the target object, specified by the dn value (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more then one attribute name by separating each name with a comma ",".

Elements of combo-box

item

Parent

page-snapin

creator-snapin

Example

```
<combo-box
  name="combobox3 "
  label="Member of:"
  attr-name="Group Membership"
  attr-type="dn"
  sync-target="Member, Equivalent To Me"
  sync-source="Security Equals"
  enabled="true"
  editable="false"
>
  <item value="Group1.Groups.NCS"/>
  <item value="Group2.Groups.NCS"/>
  <item value="Group3.Groups.NCS"/>
  <item value="Group4.Groups.NCS"/>
</combo-box>
```

dn-panel

Specifies a dn-panel input element.

Description

Specifies a input element to handle dn attributes.

Definition

```
<!ELEMENT dn-panel EMPTY>
<!ATTLIST dn-panel
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  editable (true | false) "true"
  required (true | false) "false"
  multi-valued (true | false) "false"
  rows CDATA "-1"
  attr-name CDATA #REQUIRED
  filter CDATA #IMPLIED
  sync-source CDATA #IMPLIED
  sync-target CDATA #IMPLIED
  show-name-only (true | false) "false"
  start-context CDATA #IMPLIED
  action CDATA #IMPLIED
>
```

Attributes of dn-panel

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

multi-valued

Specifies if this field handles a multi valued attribute or not. . Can be only "true" or "false". If not set, it defaults to "true".

rows

Specifies the amount of rows displayed in this <dn-panel>.

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

filter

This can only be set if "attr-type" is set to "dn". It specifies the object class to be selected in the object browser. You can enter more than one class name by separating each name with a comma ",".

sync-source

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the object you are editing (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

sync-target

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the target object, specified by the dn value (e.g. "Group Membership" and "Security

Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

show-name-only

This attribute lets you specify whether you want to show the whole dn or only the naming part (=object name). Values can be "true" or "false". Default is "false".

start-context

This attribute lets you set the initial context where the object entry browser positions itself when it opens. You could e.g. point it to the groups container in your tree. Use a dotted dn; e.g.: "ou2.ou1.o".

action

This lets you execute an Advanced Snapin command.

Elements of dn-panel

None.

Parent

page-snapin

creator-snapin

Example

```
<dn-panel
  name="DNPanel1"
  label="Member of:"
  attr-name="Group Membership"
  rows="5"
  sync-target="Member, Equivalent To Me"
  sync-source="Security Equals"
  filter="Group" />
<dn-panel
  name="DNPanel2"
  label="Security Equals:"
  attr-name="Security Equals"
  rows="5"
  editable="false" />
```

item

Specifies a label input element.

Description

Specifies a label input element.

Definition

```
<!ELEMENT item EMPTY>
<!ATTLIST item
  value CDATA #IMPLIED
>
```

Attributes of item

value

Holds the value of the item element.

Elements of item

None.

Parent

combo-box

Example

```
<combo-box
  name="combobox1 "
  label="Login Disabled:"
  attr-name="Login Disabled"
  attr-type="state"
  editable="false">
  <item value="false"/>
  <item value="true"/>
</combo-box>
```

label

Specifies a label input element.

Description

Specifies a label input element. It can be used to set titles or to enter empty lines for layout reasons.

Definition

```
<!ELEMENT label EMPTY>
<!ATTLIST label
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  text CDATA #IMPLIED
  action CDATA #IMPLIED
>
```

Attributes of label

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

text

Specifies the label's text.

action

This let's you execute an Advanced Snapin command.

Elements of label

None.

Parent

page-snapin

creator-snapin

Example

```
<label name="Label14" text=" "/>
<label name="Label15" text="Some RadioButton Examples: "/>
```

picture

Specifies a picture input element.

Description

It allows you to display pictures stored in NDS in a GIF or JPG/JPEG format.

Definition

```
<!ELEMENT picture EMPTY>
<!ATTLIST picture
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  editable (true | false) "true"
  required (true | false) "false"
  width CDATA "-1"
  height CDATA "-1"
  resizing (true | false) "true"
  attr-name CDATA #REQUIRED
  attr-type (string) "string"
>
```

Attributes of <picture>

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true". Is actually the same as "editable".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true". Is actually the same as "enabled".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

width

Specifies the width of the picture input element (the picture will not be resized) in pixels.

height

Specifies the height of the picture input element (the picture will not be resized) in pixels.

resizing

Specifies whether the picture input element should resize itself to the picture's size.

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

Elements of picture

None.

Parent

page-snapin

creator-snapin

Example

```
<picture
  name="picture1"
  attr-name="photo"
  label="Picture: "
  width="110"
  height="140"
  resizing="true"/>
```

radio-button

Specifies a radio button input element.

Description

Specifies a radio button input element.

Definition

```
<!ELEMENT radio-button EMPTY>
<!ATTLIST radio-button
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  attr-name CDATA #REQUIRED
  attr-type (string | state | dn | int) "string"
  text CDATA #IMPLIED
  radio-value CDATA #REQUIRED
  action CDATA #IMPLIED
>
```

Attributes of radio-button

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

text

Set's the text to be displayed to the right of the input element.

Radio-value

Sets the value of this radio button input element. This is compared to the attributes value and than the radio button is set to selected or not.

sync-source

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the object you are editing (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

sync-target

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the target object, specified by the dn value (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

action

This lets you execute an Advanced Snapin command.

Elements of radio-button

None.

Parent

page-snapin

creator-snapin

Example

```
<radio-button
  name="RadioButton1 "
  enabled="true "
  attr-name="Login Disabled"
  label="Login disabled:"
  text="yes"
  radio-value="true"/>
<radio-button
  name="RadioButton2 "
  enabled="true "
  attr-name="Login Disabled"
  label="Login disabled:"
  text="no"
  radio-value="false"/>
```

text-area

Specifies a text area input element.

Description

Specifies a text area input element. It can handle attributes of type (syntax) "string".

Definition

```
<!ELEMENT text-area EMPTY>
<!ATTLIST text-area
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  editable (true | false) "true"
  required (true | false) "false"
  multi-valued (true | false) "false"
  max-length CDATA "-1"
  columns CDATA "-1"
  rows CDATA "-1"
  attr-name CDATA #REQUIRED
  attr-type (string) "string"
  action CDATA #IMPLIED
>
```

Attributes of text-area

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

multi-valued

Specifies if this field handles a multi valued attribute or not. . Can be only "true" or "false". If not set, it defaults to "true".

max-length

Sets the maximum amount of characters for this field.

columns

Specifies the amount of displayed characters per line.

rows

Specifies the amount of displayed characters in this field.

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

action

This let's you execute an Advanced Snapin command.

Elements of text-area

None.

Parent

page-snapin

creator-snapin

Example

```
<text-area
  name="TextArea1 "
  attr-name="Description"/>
<text-area
  name="TextArea2 "
  attr-name="Description"
  max-length="20"/>
<text-area
  name="TextArea3 "
  attr-name="Description"
  max-length="192 "
  columns="64 "
  rows="3"/>
```

text-field

Specifies a text field input element.

Description

Specifies a text field input element. It can handle attributes of type (syntax) "string", "integer" or "dn".

Definition

```
<!ELEMENT text-field (default-value*)>
<!ATTLIST text-field
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  editable (true | false) "true"
  required (true | false) "false"
  multi-valued (true | false) "false"
  max-length CDATA "-1"
  columns CDATA "-1"
  attr-name CDATA #REQUIRED
  attr-type (string | state | dn | int) "string"
  sync-local CDATA #IMPLIED
  sync-remote CDATA #IMPLIED
  action CDATA #IMPLIED
>
```

Attributes of <text-field>

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

multi-valued

Specifies if this field handles a multi valued attribute or not. . Can be only "true" or "false". If not set, it defaults to "true".

max-length

Sets the maximum amount of characters for this field.

columns

Specifies the amount of displayed characters in this field.

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

attr-type

Specifies the type (syntax) of the handled eDirectory attribute.

filter

This can only be set if "attr-type" is set to "dn". It specifies the object class to be selected in the object browser. You can enter more than one class name by separating each name with a comma ",".

sync-source

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the object you are editing (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

sync-target

This can only be set if "attr-type" is set to "dn". It gives you the possibility to synchronize dn attributes of the target object, specified by the dn value (e.g. "Group Membership" and "Security Equals" in class "User" or "Member" and "Equivalent To Me" in class "Group"). You can enter more than one attribute name by separating each name with a comma ",".

action

This let's you execute an Advanced Snapin command.

Elements of text-field

None.

Parent

page-snapin

creator-snapin

Example

```
<text-field
  name="TextField1"
  label="TextField1"
  attr-name="Description"
  multi-valued="true"/>
<text-field
  name="TextField2"
  label="TextField2"
  attr-name="Description"
  max-length="20"/>
<text-field
  name="TextField3"
  label="TextField3"
  attr-name="Description"
  max-length="20"
  columns="10"/>
<text-field
  name="TextField4"
  label="Member of:"
  attr-name="Group Membership"
  attr-type="dn"
  multi-valued="true"
  sync-target="Member, Equivalent To Me"
  sync-source="Security Equals"
  filter="Group"/>
<text-field
  name="TextField5"
  label="Security Equals:"
  attr-name="Security Equals"
  attr-type="dn"
  multi-valued="true"
  editable="false"/>
```

xml-panel

Specifies an XML input element.

Description

Specifies a input element to handle attributes of type (syntax) "stream that contain XML data. This element is not an XML editor but a possibility to easily edit XML documents of a specific format.

Definition

```
<!ELEMENT xml-panel EMPTY>
<!ATTLIST xml-panel
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  label CDATA #IMPLIED
  enabled (true | false) "true"
  editable (true | false) "true"
  required (true | false) "false"
  attr-name CDATA #REQUIRED
  action CDATA #IMPLIED
>
```

Attributes of xml-panel

name

(Invisible) but required name of the input element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

label

Label to be displayed on the left side of the input element.

enabled

Sets the state of the input element. Can be only "true" or "false". If not set, it defaults to "true".

editable

Specifies if the user can edit the contents of the text field. . Can be only "true" or "false". If not set, it defaults to "true".

required

Specifies if the field has to contain a value or not. . Can be only "true" or "false". If not set, it defaults to "false".

attr-name

Specifies the name of the eDirectory attribute to be handled by this input element.

action

This let's you execute an Advanced Snapin command.

Elements of xml-panel

None.

Parent

page-snapin

creator-snapin

Example

```
<xml-panel  
  name="XmlPanel1 "  
  label=" "  
  attr-name="XmlData "  
>
```

Special Elements

border

Specifies a border element.

Description

Specifies an element that has no functionality but is for design purposes only. It draws a border following the constraints given. Optionally a text can be used to describe the border content.

Definition

```
<!ELEMENT border EMPTY>
<!ATTLIST border
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  text CDATA #IMPLIED
>
```

Attributes of border

name

(Invisible) but required name of the element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

text

Specifies the border's text.

Elements of border

None.

Parent

page-snapin

creator-snapin

Example

```
<border
  name="Border1 "
  text="border label "
/>
```

button

Specifies a button element.

Description

Buttons can be used to call functions or plug-ins.

Definition

```
<!ELEMENT button EMPTY>
<!ATTLIST button
  name CDATA #REQUIRED
  constraints CDATA #IMPLIED
  enabled (true | false) "true"
  text CDATA #IMPLIED
  action CDATA #IMPLIED
>
```

Attributes of button

name

(Invisible) but required name of the element. This name must be unique across the whole snapin definition.

constraints

Contains the rectangle (x,y,width,height) of the element, e.g.: *constraints="0,0,210,25"*. Takes only effect with *layout="absolute"*.

enabled

Sets the state of the element and can only be "true" or "false". If not set, it defaults to "true".

text

Specifies the button text.

action

This let's you execute an Advanced Snapin command or plug-in. A button's action is only executed if the button is pushed.

Elements of button

None.

Parent

page-snapin

creator-snapin

Example

```
<button
  name="Button1 "
  constraints="Button1"
```

```

    text="Create Full Name"
    action="setValue('tfFullName', concat(getValue('tfGivenName'), ' ',
    getValue('tfSurname')))"
  />

```

before-creation-action

Specifies a before-creation-action element.

Description

Specifies an action element that is executed before the object is being created. This element can only be used with creator snap-ins.

Definition

```

<!ELEMENT before-creation-action EMPTY>
<!ATTLIST before-creation-action
  name CDATA #REQUIRED
  action CDATA #IMPLIED
>

```

Attributes of before-creation-action

name

(Invisible) but required name of the element. This name must be unique across the whole snapin definition.

action

This let's you execute an Advanced Snapin command.

Elements of before-creation-action

None.

Parent

creator-snapin

Example

```

<before-creation-action
  name="Action01"
  action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.NCSUNG',conca
t(getForeignValue('name'),'10.0.1.3:389', 'true'))"
/>

```

after-creation-action

Specifies a before-creation-action element.

Description

Specifies an action element that is executed after the object has been created. This element can only be used with creator snap-ins.

Definition

```
<!ELEMENT after-creation-action EMPTY>
<!ATTLIST after-creation-action
  name CDATA #REQUIRED
  action CDATA #IMPLIED
>
```

Attributes of after-creation-action

name

(Invisible) but required name of the element. This name must be unique across the whole snapin definition.

action

This let's you execute an Advanced Snapin command.

Elements of after-creation-action

None.

Parent

creator-snapin

Example

```
<after-creation-action
  name="Action01"
  action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.NCSUNG',concat(getForeignValue('name'),'10.0.1.3:389', ' ',true'))"
/>
```

Supported Syntax Mapping

To not make things confusing, Advanced Snapin uses the same syntax mapping as DirXML does. So the following is an excerpt from the Novell Developer Kit "DirXML Drivers":

"...NDS supports 28 syntax definitions that can be used for an attribute's type. DirXML supports only 10. The following table shows how the 28 definitions have been mapped to the 10 definitions and the component names that have been supplied for the various structured syntax definitions.

NDS Syntax Name	XML Type	Components and Notes	Supported?
SYN_BACK_LINK	structured	two components: - serverDn (referential) - remoteld	No
SYN_BOOLEAN	state	"true" or "false"	Yes
SYN_CE_STRING	string		Yes
SYN_CI_LIST	structured	one or more string components	Yes
SYN_CI_STRING	string		Yes
SYN_CLASS_NAME	className		No
SYN_COUNTER	Counter		No
SYN_DIST_NAME	dn	referential	Yes
SYN_EMAIL_ADDRESS	structured	two components: - eMailType - eMailAddr	Yes
SYN_FAX_NUMBER	string (actually, this is a structured type, but enable it right now, it is mapped to a string.)	three components: - faxNumber - faxBitCount - faxParameters base64 encoded data	Yes
SYN_HOLD	structured	two components: - holdEntryDn (referential) - holdAmount	No
SYN_INTEGER	Int		Yes
SYN_INTERVAL	interval		No
SYN_NET_ADDRESS	structured	two components - netAddrType - netAddr base64 encoded data	No
SYN_NU_STRING	string		Yes
SYN_OBJECT_ACL	structured	three components - protectedName - trustee (referential) - privileges	No
SYN_OCTET_LIST	structured	one or more octet components.	No

		base64 encoded data	
SYN_OCTET_STRING	octet	base64 encoded data	Yes
SYN_PATH	structured	three components: - nameSpace - volume (referential) - path	No
SYN_PO_ADDRESS	Structured	6 string components	No
SYN_PR_STRING	string		Yes
SYN_REPLICA_POINTER	structured	four components - server (referential) - replicaType - replicaNumber - count	No
SYN_STREAM	octet	Base64 encoded data	Yes
SYN_TEL_NUMBER	string (actually, this is a teleNumber type, but enable it right now, it is mapped to a string.)		Yes
SYN_TIME	time		Yes
SYN_TIMESTAMP	structured	three components: - seconds - replicaNumber - eventID	No
SYN_TYPED_NAME	structured	three components: - dn (referential) - level - interval	No
SYN_UNKOWN	octet	base64 encoded data	No

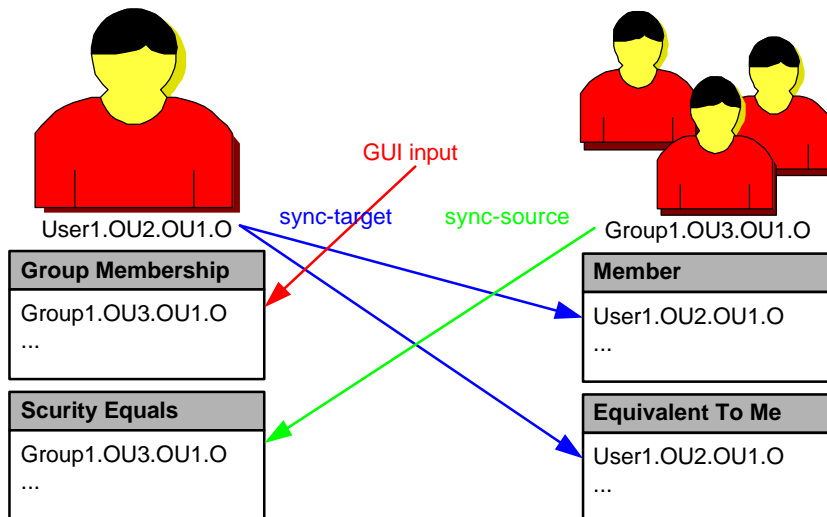
Synchronization

Synchronization allows you to manage attributes of type (syntax) "dn" in a highly flexible way. The may be best known example is the management of user and group memberships and their corresponding security attributes.

The following code snippet is used to handle the user part:

```
<text-field
  name="TextField"
  label="Member of:"
  attr-name="Group Membership"
  attr-type="dn"
  multi-valued="true"
  sync-target="Member, Equivalent To Me"
  sync-source="Security Equals"
  filter="Group"/>
```

Synchronization



Actions - Functions

Overview

Actions greatly enhance the functionality of an Advanced Snapin. An action consists of one or multiple nested functions that interact with other fields of your snap in, attributes or plug ins. An action could e.g. combine the values of "Given Name" and "Surname" and automatically fill in the "Full Name" field.

Functions

Syntax

```
result = name(parameter1, parameter2)
```

```
'James Bond' = getValue('TextField1')
```

result

The result of each and every function call is always a string containing zero or more characters.

name

The name of a function gives you an idea of the function is actually doing. The first letter of a function usually is lower case. An opening bracket immediately follows the name of a function.

parameters

Each and every parameter is basically a string containing zero or more characters. Commas might separate multiple parameters. Brackets surround all parameters of a function. A parameter string might be constructed in two different ways:

1. As a quoted fixed string: `'this is a fixed string parameter'`
2. As a function. The result of this function is then taken as the parameter: `getValue('TextField1')`

This allows you to nest as many functions as you need to get the desired result.

Function Index

name	variations	result	parameters	description / sample
if	if(<condition>, <if true>)	result of functions inside the if() function as a string	<condition>: expression that returns "true" or "false". can be function or plugin	Allows a conditional action: The expression <if true> is only executed if <condition> returns "true".
	if(<condition>, <if true>, <if false>)		<if true>: branch to follow when condition returns true <if false>: branch to follow when condition returns false	Allows a very simple branching: The expression <if true> is only executed if <condition> returns "true", <if false> is only executed if <condition> returns "false".
getValue	getValue()	value as a string	<input element>: name of an input element	Gets the value of the input element this action belongs to.
	getValue(<input element >)		<trigger>: "true" or "false". Specifies whether changes in the target input element should be triggered or not	Gets the value of the input element that is referred to by <input element>.
	getValue(<input element >, <trigger>)			Gets the value of the input element that is referred to by <input element> and triggers for changes.
setValue	setValue(<value>)	value as a string	<value>: value as a string	Sets the value of the input element this action belongs to.
	setValue(<input element>, <value>)		<input element>: name of an input element	Sets the value of the input element that is referred to by <input element>.
concat	concat(<p1>, <p2>, ...)	concatenated values as a string	two or more strings	Concatenates two or more strings.
callPlugin	callPlugin(<classname>, <params>)	result of the plug-in as a string	<classname>: name of the plug-in's class including package <params>: the plug-in parameters	Calls a plug-in: <code>callPlugin('com.novell.ncs.consoleone.advanced.plugins.MyFirstPlugin', 'blabla')</code> (refer to chapter "Plug-Ins" for more info)
hasValue	hasValue()	"true"/"false"	<input element>: name of an input element	checks if the input element this action belongs to has set a value or not.
	hasValue(<input element>)	"true"/"false"		checks if the input element that is referred to by <input element> has set a value or not.
isEmpty	isEmpty()	"true"/"false"	<input element>: name of an input element	checks if the input element this action belongs to is empty or not.

	isEmpty(<input element>)	"true"/"false"		checks if the input element that is referred to by <input element> is empty or not.
getAttrValue	getAttrValue(<attr>)	string	<attr>: name of an eDirectory attribute	Gets the value of the specified attribute of the current object.
	getAttrValue(<dn>, <attr>)	string	<dn>: distinguished name of an eDirectory object (name.ou.ou.o)	Gets the value of the specified attribute of the object that is referred to by <dn>.
getNewObjName	getNewObjName()	string	-	Returns the name of the new object. <i>Note: This function is for use with creator snap-ins only.</i>
getForeignValue	getForeignValue(<name>)	string	<name>: name of the component.	Gets the value of a foreign component. E.g. a text field from another snap-in. <i>Note: Use the NCSGuiSpy plug-in to "spy" for other component's name.</i>
setForeignValue	setForeignValue(<name>, <value>)	string	<name>: name of the component. <value>: value to set the component to.	Sets the value of a foreign component. E.g. a text field from another snap-in. <i>Note: Use the NCSGuiSpy plug-in to "spy" for other component's names.</i>

Sample Actions

action="if(isEmpty(), setValue(concat(getValue('TextField1'), ' ', getValue('TextField2'))))"

action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.MyFirstPlugin', 'blabla')"

Plug-Ins

Overview

Plug-ins extend the base functionality of Advanced Snapin. Plug-ins can be written using the Advanced Snapin API. This API has been developed to allow quick and optional adoptions of the base functionality of Advanced Snapin without changing the base code.

For more information please take a look at the API Java documentation and the included examples.

How to Execute Plug-Ins

A plug-in is always executed from an action attribute by calling it with the function *callPlugin* (see chapter "Actions - Function" for a complete function index), e.g.:

```
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.MyFirstPlugin', 'p1,p2,p3')"
```

the *callPlugin* function takes 2 parameters: The first specifies the plug-in's Java package and class and the second one is a parameter string. It's up to the plug-in developer how the parameter string has to look like but normally it is a comma separated string of parameters. Because Advanced Snapin has its own command interpreter, you can also construct your parameter string to be calculated at run time using other function calls:

```
action="callPlugin('com.novell.ncs.idsrv.plugin.IDSRC', concat('localhost,1099,', getValue('TextField9'), ',IDSRC,0'))"
```

In the above example, the parameter string is being concatenated at run time and contains a variable from a text field input element.

Plug-In Index

MyFirstPlugin

Package and Class	com.novell.ncs.consoleone.advanced.plugins.MyFirstPlugin
Parameter String(s)	<p1>,<p2>,...,<pn>
Return Value	Returns the string: "MyFirstPlugin successfully executed".
Developer	Volker Scheuber
Description	Show what a plug-in can do. When called, a message box will open and show some run time information and the list of parameters the plug-in was called with.

Check Unique Attribute

Package and Class	com.novell.ncs.consoleone.advanced.plugins.NCSCkAttr
--------------------------	--

Parameter String(s)	<ol style="list-style-type: none"> 1. <attribute name>, <value to check> 2. <attribute name>, <value to check>, <LDAP server and port> 3. <attribute name>, <value to check>, <LDAP server and port>, <userDN>, <password> 4. <attribute name>, <value to check>, <LDAP server and port>, <userDN>, <password>, <threshold> 5. <attribute name>, <value to check>, <LDAP server and port>, <userDN>, <password>, <threshold>, <basedn> 6. <attribute name>, <value to check>, <LDAP server and port>, <userDN>, <password>, <threshold>, <basedn>, <exclAttr>, <exclVal>
Return Value	"true" or "false"
Developer	Volker Scheuber
Description	<p>This plug-in checks a nattribute's value for uniqueness. If the name is not unique, shows a message and returns "false", "true" otherwise. You may use this plug-in in a creator snap-in to only allow object creation if the CN or any other attribute is unique or it may be used in a page snap-in to check whether the given eMail address is still available.</p> <ul style="list-style-type: none"> • attribute name: Which attribute to check for uniqueness. • value to check: Which value to check against. • LDAP server and port: LDAP server and (optionally) port, e.g.: 10.0.0.32:389. If not specified it defaults to "localhost:389". • userDN: Login name (LDAP DN). If not specified, the plug-in will do an anonymous bind. • password: Password. If not specified, the plug-in will do an anonymous bind. • threshold: How many results are allowed before "false" is returned. If not specified it defaults to "0". • basedn: Where to start the check. If not specified it defaults to root. • exclAttr: Attribute to build exclude query. This can be used (only in conjunction with exclVal) to exclude all results where exclAttr=exclVal. If not specified, no exclusion occurs. • exclVal: Attribute to build exclude query. This can be used (only in conjunction with exclAttr) to exclude all results where exclAttr=exclVal. If not specified, no exclusion occurs.

Unique Name Generator

Package and Class	com.novell.ncs.consoleone.advanced.plugins.NCSUNG
Parameter String(s)	<ol style="list-style-type: none"> 1. <value to check>, <LDAP server and port> 2. <value to check>, <LDAP server and port>, <autocorrect>
Return Value	"true" or "false".
Developer	Volker Scheuber
Description	<p>This plug-in checks a new object's name for uniqueness. The plug-in can be initialized in 2 modes:</p> <ol style="list-style-type: none"> 1. Auto Correction If the name is not unique, it adds an underscore followed by a counter (e.g.

	<p>JBond_1).</p> <p>2. Notification only If the name is not unique, show a message and vetoe creation.</p> <p>The checks are being performed using an LDAP query. Use this plug-in to automatically generate unique names by adding a counter or to just display a message if not unique.</p> <ul style="list-style-type: none"> value to check: ss LDAP server and port: ss autocorrect: If set to "true": If the name is not unique, it adds an underscore followed by a counter (e.g. JBond_1). If set to "false", only a message is displayed. If not specified, it defaults to "false".
--	---

GUI Spy

Package and Class	com.novell.ncs.consoleone.advanced.plugins.NCSGuiSpy
Parameter String(s)	Empty string or just something. The parameter is not needed but necessary for the callPlugin function.
Return Value	Returns the string: "NCSGuiSpy successfully executed".
Developer	Volker Scheuber
Description	A helper plug-in to "spy" out a java GUI for input elements. Information from this plug-in can then be used to access these foreign input elements using actions or other plug-ins.

ID Server Client

Package and Class	com.novell.ncs.idsrv.plugin.IDC
Parameter String(s)	<IP>,<RMI port>,<policy>,<client name>,<trace level>
Return Value	next ID according to the policy <policy>.
Developer	Volker Scheuber
Description	The ID Server Client may be run from the command line or from a style sheet in a DirXML driver configuration or in any Java application as an API. The ID Server Client only supports one single method and functionality: getNextID(). A call to this method or a call to the class' main() method returns the next ID from the ID Server.

ID Server Remote Control

Package and Class	com.novell.ncs.idsrv.plugin.IDSRC
Parameter String(s)	<IP>,<RMI port>,<command>,<client name>,<trace level>
Return Value	Response from ID server.
Developer	Volker Scheuber
Description	The ID Server Remote Control allows taking remotely control over the ID Server. You can issue commands, re-initialize the server or change it's operational parameters at run time.

Dynamic Combo Box

Package and Class	com.novell.ncs.consoleone.advanced.plugin.MyNextPlugin
Parameter String(s)	<container in dotted format, e.g. LOCATIONS.HRDATA>
Return Value	Not used
Developer	Rob Rawson
Description	<p>The purpose of this plug-in is to fill the combo box for the location field with the locations from HRDATA locations. The parameter to be passed in to this class is the context in dotted notation of the location container, which is normally "LOCATIONS.HRDATA".</p> <p>Changes:</p> <p>11/8/02: This class was upgraded to improve performance by removing calls to JNDI which were returning the containers in any order and then passing them through a merge sort. The new code makes internal ConsoleOne calls which return the list in sorted order removing the need for the additional sort.</p> <p>11/15/02: Correction made for blank attributes, possible duplicate entries and to assure that the current value is filled in properly.</p>

Convert Leaf to Root Most

Package and Class	com.novell.ncs.consoleone.advanced.plugin.leaf2RootMost
Parameter String(s)	String to truncate before conversion. Used to ignore part of the name being converted.
Return Value	Not used (modifies control directly)
Developer	Rob Rawson
Description	This takes a context in leafmost dotted form and changed the associated control to rootmost slashed form. The parameter is a string which should be excluded from the conversion.

Open Property Page

Package and Class	com.novell.ncs.consoleone.advanced.plugin.openPropertyPage
Parameter String(s)	<context in rootmost slashed form (starting with the tree name)>
Return Value	Not used
Developer	Rob Rawson
Description	This takes a context in rootmost slashed form (starting with the tree name) and opens the property page of the identified object

amHeritage

Package and Class	com.novell.ncs.consoleone.advanced.plugin.amHeritage
Parameter String(s)	None
Return Value	Not used (replaces control)
Developer	Rob Rawson
Description	When associated with a multi-valued text field, this creates a list box containing all of the values within that list. When associated with a button, this will open the property page of the object which is identified in the list box (rootmost slashed notation preceded by tree

	name). If the button text is "Parent", it will open the parent object to the object which is identified.
--	--

parseMe

Package and Class	com.novell.ncs.consoleone.advanced.plugin.parseMe
Parameter String(s)	Character 0 of the parameter is the separator Character 1 of the parameter is a single digit indicating which token to parse the string up to Characters 2+ contain the string to be parsed
Return Value	Not used (modifies control directly)
Developer	Rob Rawson
Description	This is associated with a label. The label acts as a parameter; the first character is the token delimiter, the second is the number of levels. The parameter of the class contains the number of levels to return. For example, /1 means the delimiter is "/" and up through the first token should be returned. If more levels are specified than exist in the parameter string, the string " - n/a - " is returned.

parseMeToo

Package and Class	com.novell.ncs.consoleone.advanced.plugin.parseMeToo
Parameter String(s)	String to be parsed
Return Value	Not used (modifies control directly)
Developer	Rob Rawson
Description	This is associated with a label. The prior value of the label is the delimiter, the parameter for the class contains a string to be parsed, the label is changed to the value of the last token in the parameter string.

Selection List Control

Package and Class	com.novell.ncs.consoleone.advanced.plugin.vcPagePlugin
Parameter String(s)	Tree Name/Context (in dotted notation). The value of the attribute associated with this control is used to populate the left list box, the OU's within this container will be used to populate the right list box with any values from the left box excluded.
Return Value	Not used (replaces control)
Developer	Rob Rawson
Description	When associated with a multi-valued text field, this class hides the control and replaces that control with two list boxes. The left box will contain what would have been the values within the text field. The parameter contains a string indicating the context to read child containers from which will contain the values to place in the right list. Any value in the left list is automatically excluded from the right list. When used as an action on a button with text "<<", this moves the selected entry out of the right list and into the left one. When used as an action on a button with text ">>", this moves the selected entry out of the left list and into the right one.

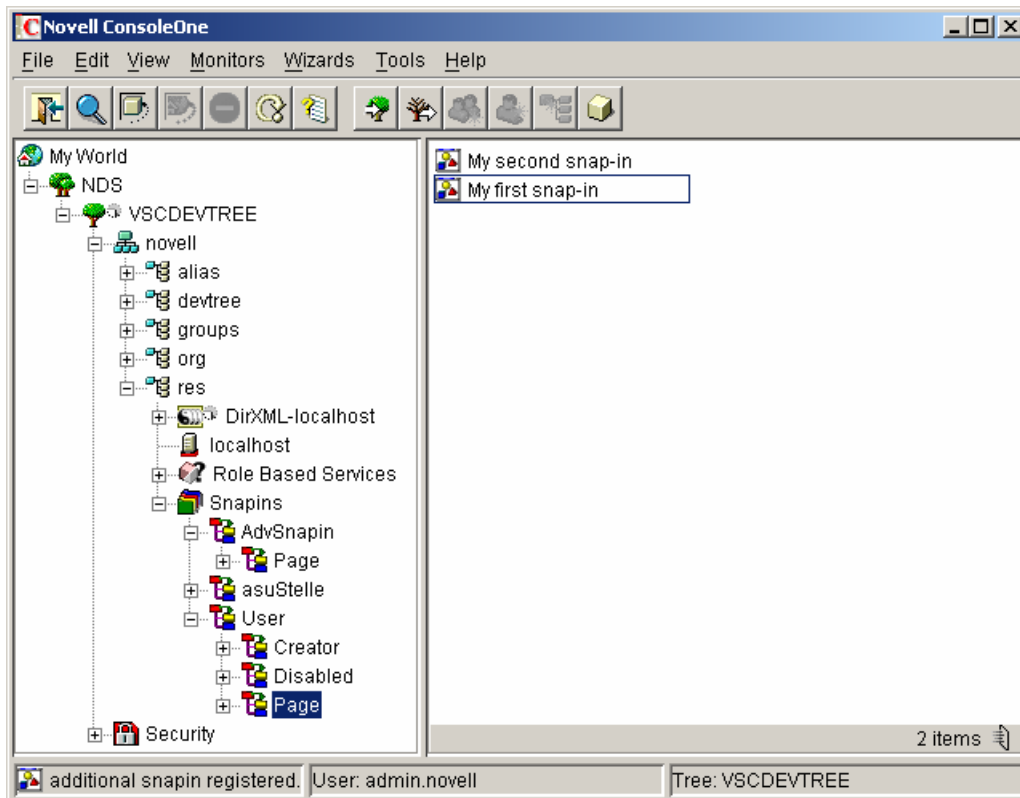
Sample Snap-Ins

My first snap-in

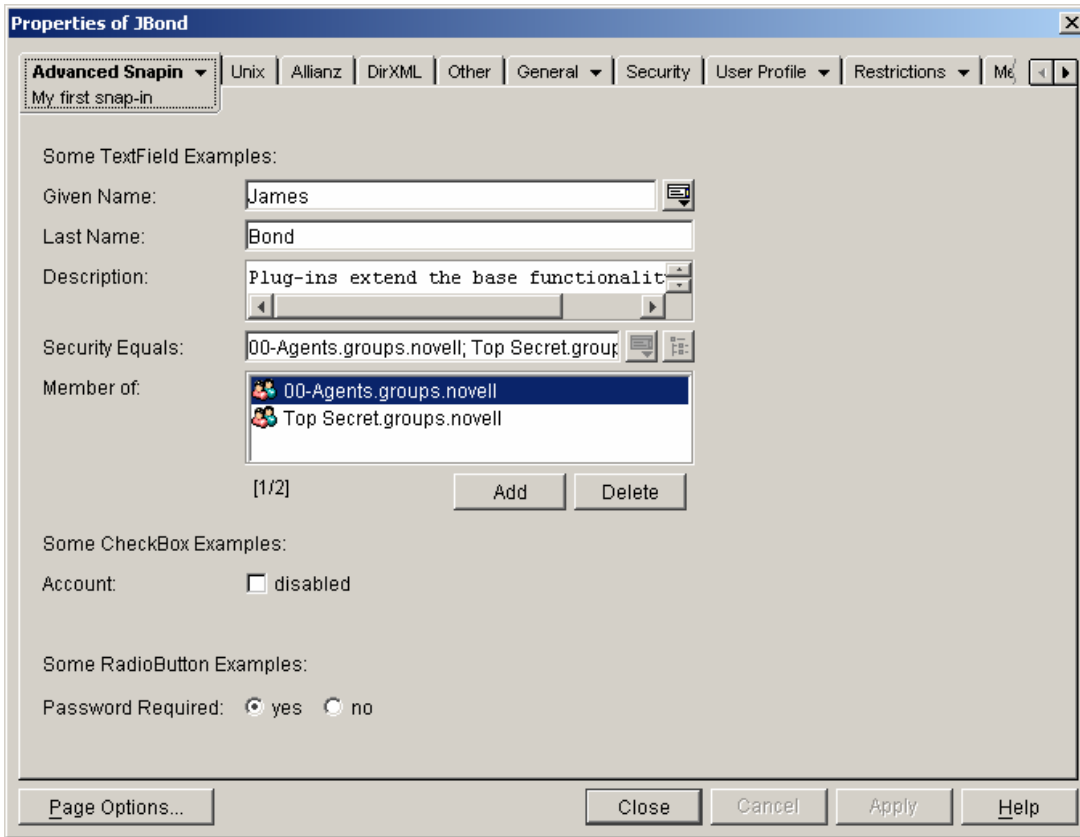
XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.0.7 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="MyFirstSnapin" snapin-description="My first snap-in" snapin-tab-name="Advanced Snapin"
  snapin-menu-name="My first snap-in">
    <label name="Label1" text="Some TextField Examples:"/>
    <text-field name="TextField1" label="Given Name:" attr-name="Given Name" multi-valued="true"/>
    <text-field name="TextField2" label="Last Name:" attr-name="Surname" max-length="20"/>
    <text-area name="TextAreal" label="Description:" attr-name="Description" rows="2"/>
    <text-field name="TextField4" label="Security Equals:" attr-name="Security Equals" attr-type="dn" multi-
  valued="true" editable="false"/>
    <dn-panel name="DNPanel1" label="Member of:" attr-name="Group Membership" attr-type="dn" rows="3" sync-
  target="Member, Equivalent To Me" sync-source="Security Equals" filter="Group"/>
    <label name="Label3" text="Some CheckBox Examples:"/>
    <check-box name="CheckBox1" attr-name="Login Disabled" attr-type="boolean" label="Account:" text="disabled"
  radio-value="true"/>
    <label name="Label4" text=""/>
    <label name="Label5" text="Some RadioButton Examples:"/>
    <radio-button name="RadioButton1" enabled="true" attr-name="Password Required" label="Password Required:"
  text="yes" radio-value="true"/>
    <radio-button name="RadioButton2" enabled="true" attr-name="Password Required" label="Password Required:"
  text="no" radio-value="false"/>
  </page-snapin>
</adv>
```

Snap-In Objects



Screenshot of "My first snap-in" in Action



My second snap-in

```

XML Code
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.0.7 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="MySecondSnapin" snapin-description="My second snap-in" snapin-tab-
name="Advanced Snapin" snapin-menu-name="My second snap-in" layout="absolute">
    <!-- column 1 -->
    <label constraints="0,0,210,20" name="Label101" text="Given Name:"/>
    <text-field constraints="0,20,210,20" name="TextField101" attr-name="Given Name" multi-
valued="false"/>
    <label constraints="0,45,210,20" name="Label102" text="Last Name:"/>
    <text-field constraints="0,65,210,20" name="TextField102" attr-name="Surname" multi-
valued="false"/>
    <label constraints="0,90,210,20" name="Label103" text="Full Name:"/>
    <text-field constraints="0,110,210,20" name="TextField103" attr-name="Full Name"
action="setValue(concat(getValue('TextField101','true'),' ',getValue('TextField102','true')))/>
    <label constraints="0,140,60,20" name="Label104" text="Account:"/>
    <check-box constraints="65,140,160,20" name="CheckBox101" attr-name="Login Disabled" attr-
type="boolean" text="disabled" radio-value="true"/>
    <label constraints="0,165,210,20" name="Label105" text="Password Expires:"/>
    <text-field constraints="0,185,335,20" name="TextField104" attr-name="Login Expiration Time"/>
    <label constraints="0,210,210,20" name="Label106" text="Description:"/>
    <text-area constraints="0,230,335,155" name="TextArea101" attr-name="Description"/>
    <!-- column 2 -->
    <label constraints="225,0,110,20" name="Label201" text="Photo:"/>
    <picture constraints="225,20,110,140" name="picture201" attr-name="photo" width="110"
height="140" resizing="true"/>
    <!-- column 3 -->
    <label constraints="350,0,250,20" name="Label301" text="Groups:"/>
    <dn-panel constraints="350,20,250,140" name="DNPanel301" label="Member of:" attr-name="Group
Membership" attr-type="dn" rows="3" sync-target="Member, Equivalent To Me" sync-source="Security
Equals" filter="Group"/>
    <label constraints="350,165,250,20" name="Label302" text="eMail:"/>
    <text-field constraints="350,185,250,20" name="TextField301" attr-name="Internet EMail Address"
editable="true"/>
    <label constraints="350,210,250,20" name="Label303" text="Phone:"/>
    <text-field constraints="350,230,250,20" name="TextField302" attr-name="Telephone Number"/>
  </page-snapin>
</adv>

```

```

<label constraints="350,255,250,20" name="Label304" text="Fax:"/>
<text-field constraints="350,275,250,20" name="TextField303" attr-name="Facsimile Telephone
Number"/>
<label constraints="350,300,250,20" name="Label305" text="Pager:"/>
<text-field constraints="350,320,250,20" name="TextField304" attr-name="pager"/>
<label constraints="350,345,250,20" name="Label306" text="mobile:"/>
<text-field constraints="350,365,250,20" name="TextField305" attr-name="mobile"/>
</page-snapin>
</adv>

```

Screenshot of "My second snap-in" in Action

Properties of JBond

Advanced Snapin | Unix | Allianz | DirXML | Other | General | Security | User Profile | Restrictions | Me

My second snap-in

Given Name: James

Last Name: Bond

Full Name: James Bond

Account: disabled

Password Expires: February 28, 2002 10:00:00 PM GMT+01:00

Description: Plug-ins extend the base functionality of Advanced Snapin. Plug-ins can be written using the Advanced Snapin API. This API has been developed to allow quick and optional adoptions of the base functionality of Advanced Snapin without changing the base code.

Photo:

Groups: [1/2]

- 00-Agents.groups.novell
- Top Secret.groups.novell

eMail: jbond@mi6.co.uk

Phone:

Fax:

Pager:

mobile:

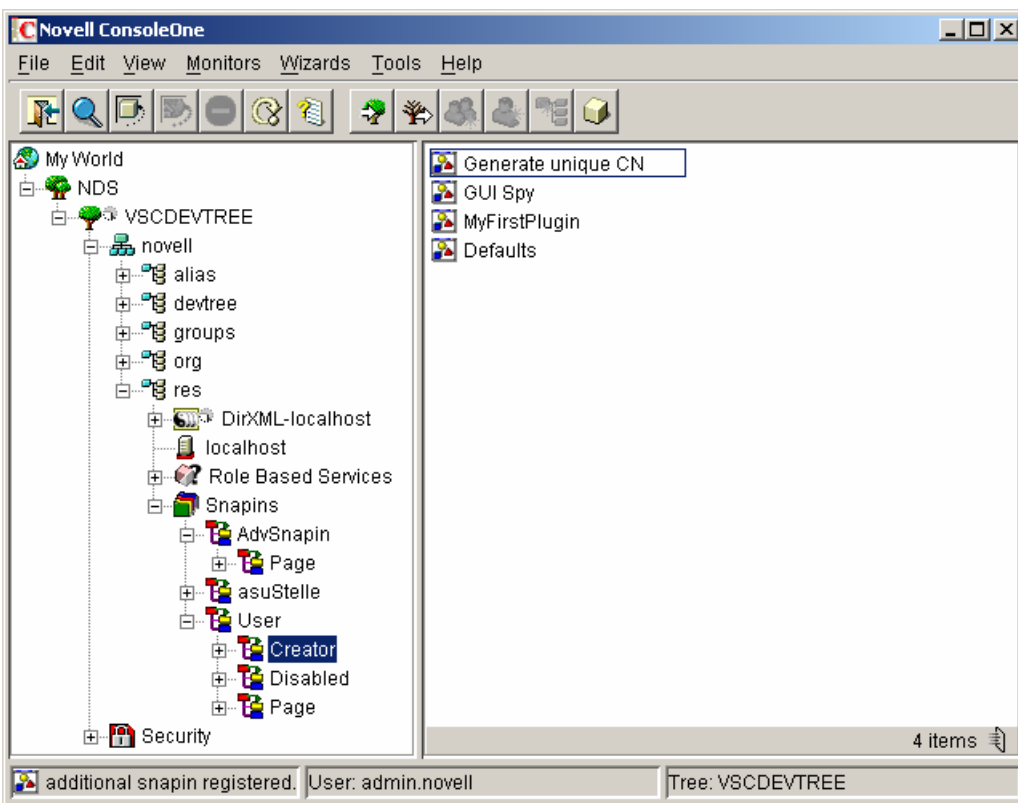
Buttons: Page Options..., Close, Cancel, Apply, Help

Generate unique CN (Creator snap-in)

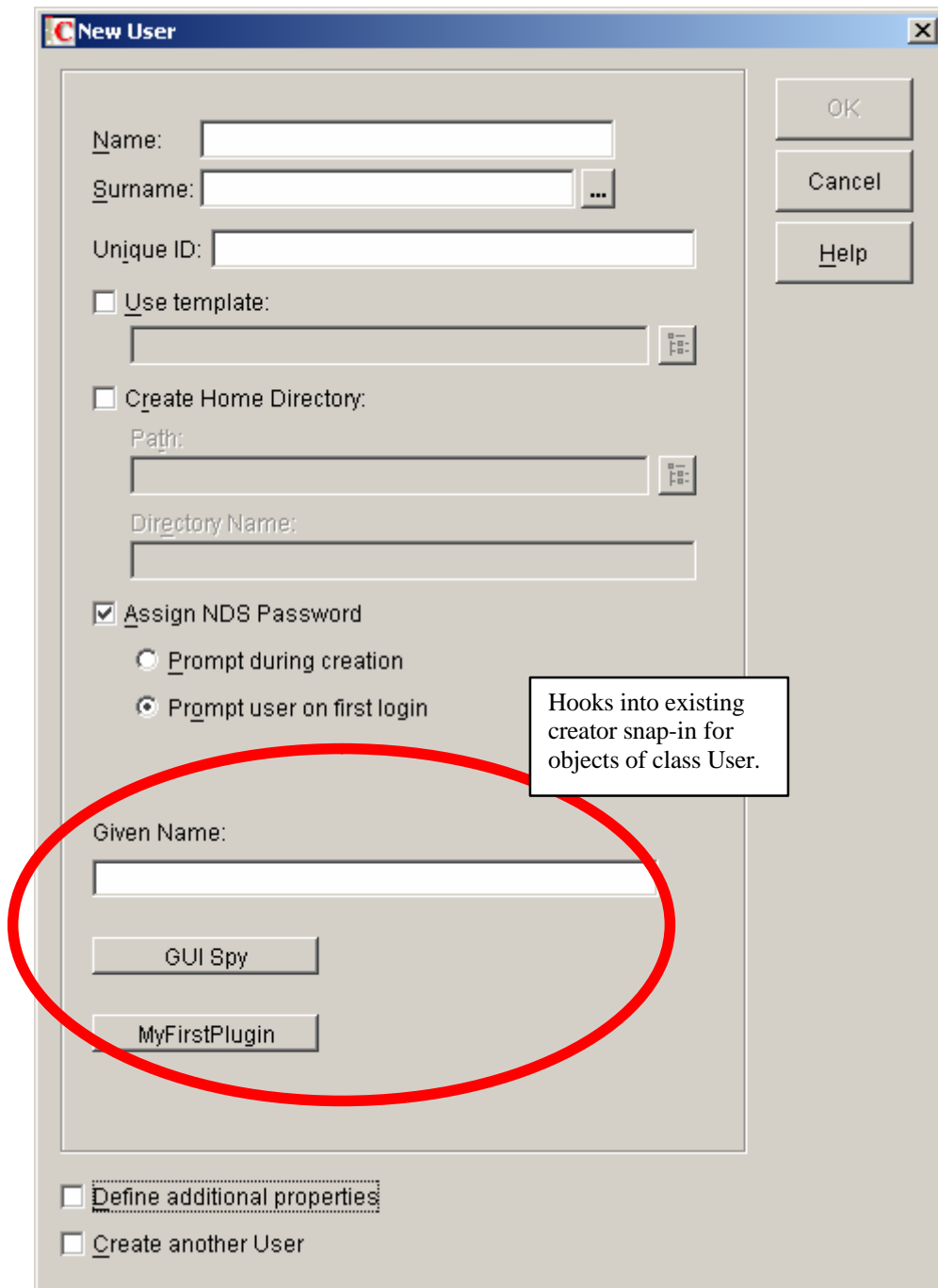
XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <creator-snapin snapin-name="NCS Unique Name Generator" snapin-description="Generate Unique CN"
  layout="absolute">
    <before-creation-action name="Action01"
  action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.NCSUNG',concat(getForeignValue('name'),' ',10.0.1.
  3:389', ',true'))"/>
    <label constraints="0,0,300,20" name="Label01" text="Given Name:"/>
    <text-field constraints="0,25,300,20" name="Textfield01" attr-name="Given Name" required="true"/>
    <text-field constraints="0,0,0,0" name="Textfield02" attr-name="Full Name"
  action="setValue(concat(getValue('Textfield01', 'true'), ' ', getForeignValue('StringEditor')))" />
  </creator-snapin>
</adv>
```

Snap-In Objects



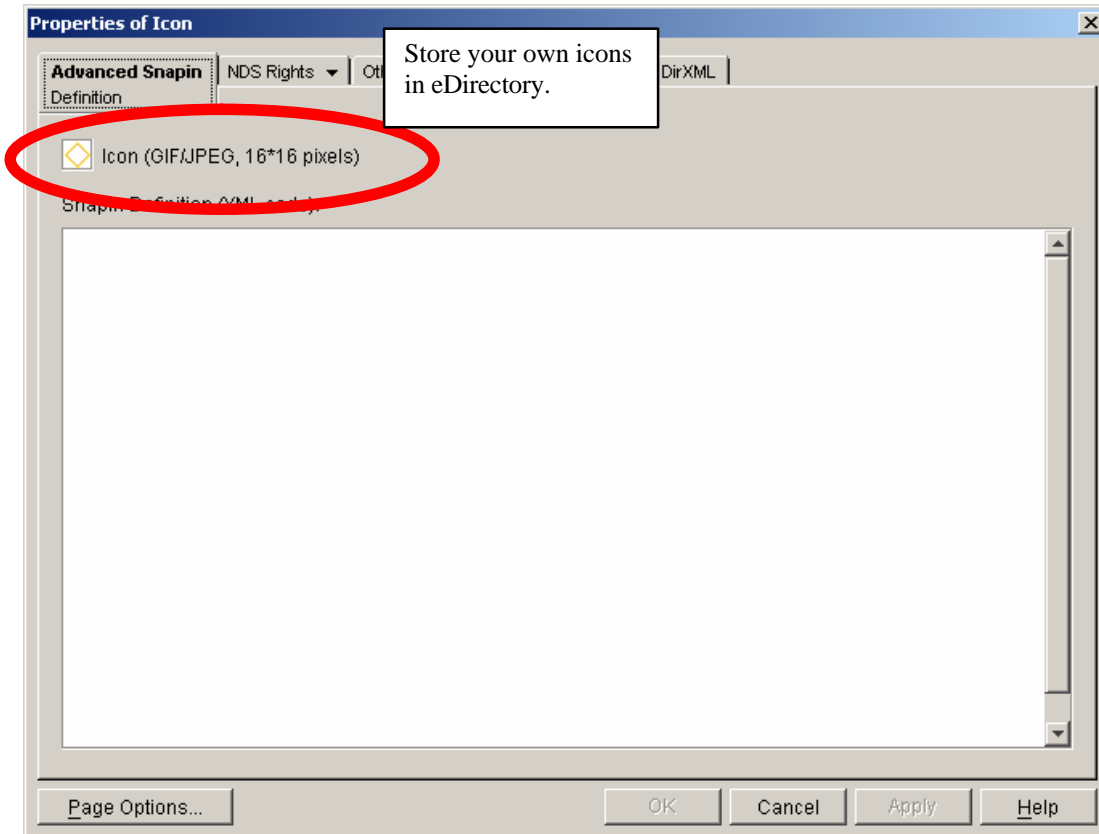
Screenshot of "myFirstCreatorSnapin" in Action



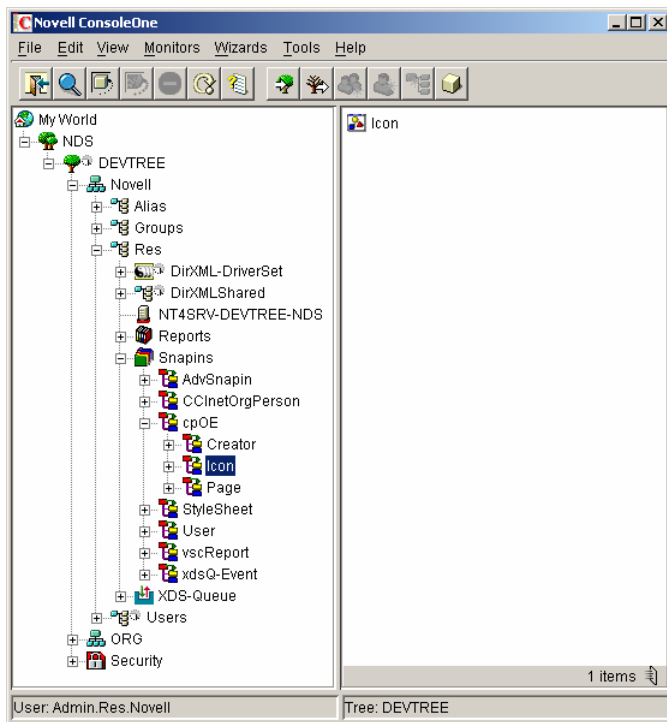
myFirstIconSnapin

XML Code / Snapin Definition

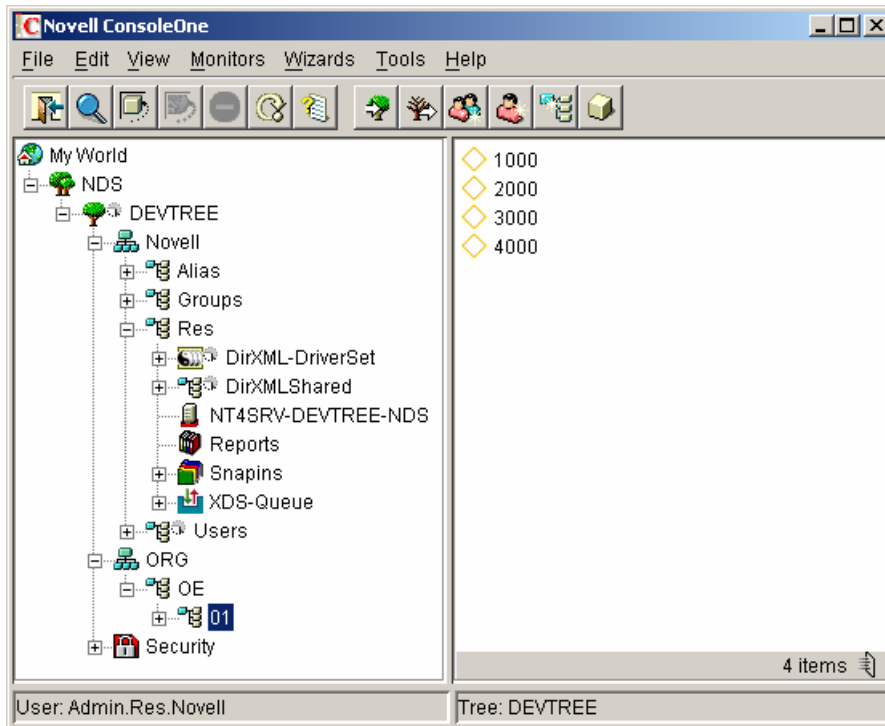
No code is needed, only an image:



Snopin Objects



Screenshot of "myFirstIconSnopin" in Action



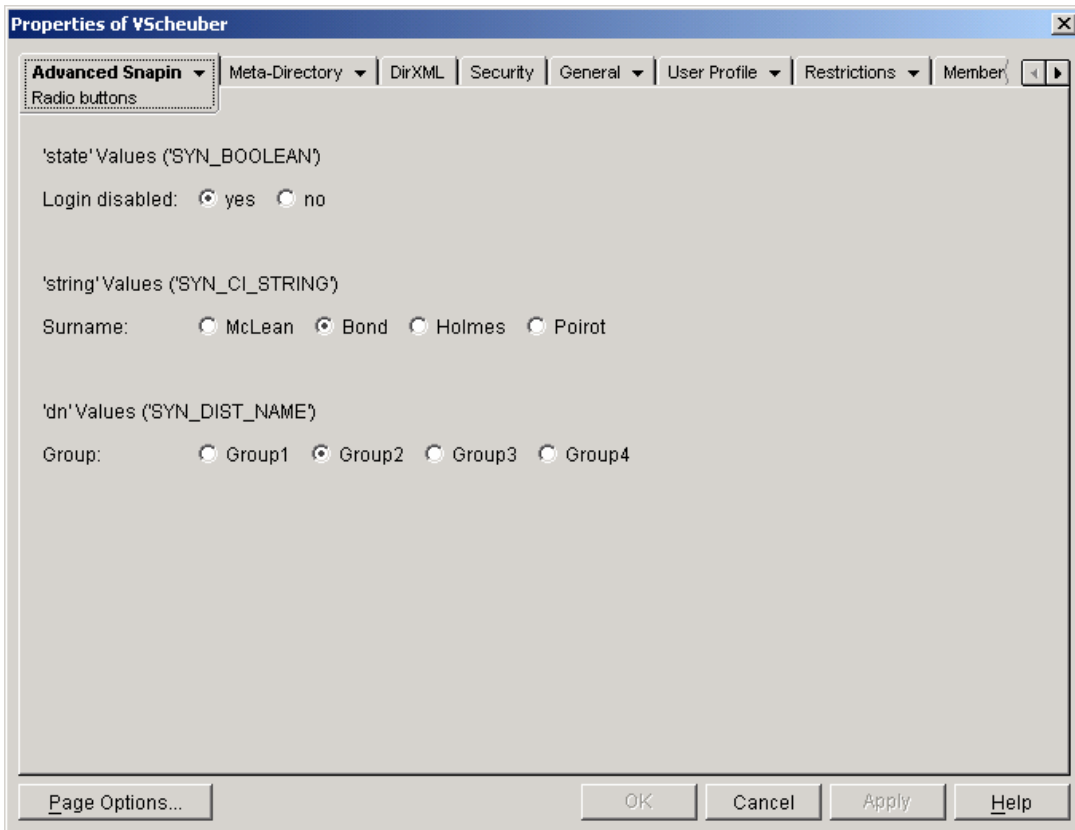
RadioButtons

```

XML Code
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="RadioButtons" snapin-description="Show the functionality of radio buttons" snapin-
  tab-name="Advanced Snapin" snapin-menu-name="Radio buttons" class="User" scope="page">
    <label name="Label1" text="'state' Values ('SYN_BOOLEAN')"/>
    <radio-button name="RadioButton1" enabled="true" attr-name="Login Disabled" label="Login disabled:"
  text="yes" radio-value="true"/>
    <radio-button name="RadioButton2" enabled="true" attr-name="Login Disabled" label="Login disabled:"
  text="no" radio-value="false"/>
    <label name="Label2" text=""/>
    <label name="Label3" text="'string' Values ('SYN_CI_STRING')"/>
    <radio-button name="RadioButton3" attr-name="Surname" label="Surname:" text="McLean" radio-
  value="McLean"/>
    <radio-button name="RadioButton4" attr-name="Surname" label="Surname:" text="Bond" radio-value="Bond"/>
    <radio-button name="RadioButton5" attr-name="Surname" label="Surname:" text="Holmes" radio-
  value="Holmes"/>
    <radio-button name="RadioButton6" attr-name="Surname" label="Surname:" text="Poirot" radio-
  value="Poirot"/>
    <label name="Label4" text=""/>
    <label name="Label5" text="'dn' Values ('SYN_DIST_NAME')"/>
    <radio-button name="RadioButton7" attr-name="Group Membership" attr-type="dn" label="Group:"
  text="Group1" radio-value="Group1.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security
  Equals"/>
    <radio-button name="RadioButton8" attr-name="Group Membership" attr-type="dn" label="Group:"
  text="Group2" radio-value="Group2.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security
  Equals"/>
    <radio-button name="RadioButton9" attr-name="Group Membership" attr-type="dn" label="Group:"
  text="Group3" radio-value="Group3.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security
  Equals"/>
    <radio-button name="RadioButton10" attr-name="Group Membership" attr-type="dn" label="Group:"
  text="Group4" radio-value="Group4.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security
  Equals"/>
  </page-snapin>
</adv>

```

Screenshot of "RadioButtons" in Action

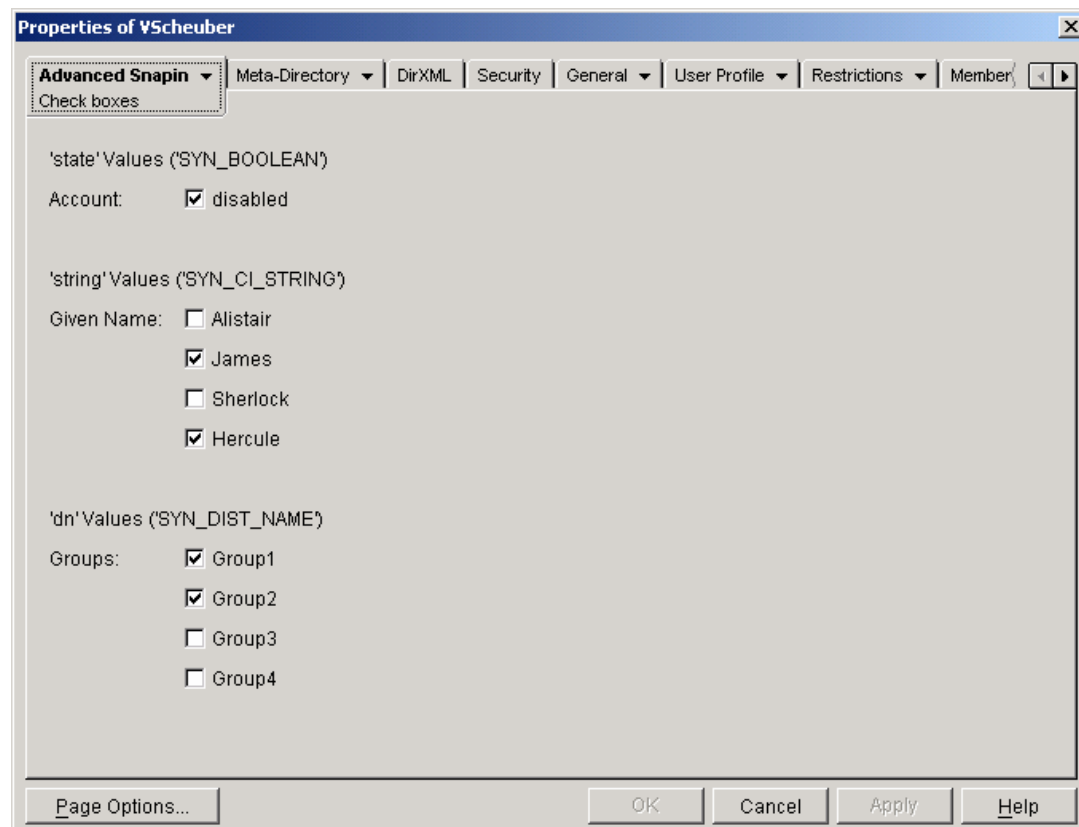


CheckBoxes

XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.0.7 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="CheckBoxes" snapin-description="Show functionality of check boxes" snapin-tab-
name="Advanced Snapin" snapin-menu-name="Check boxes" class="User" scope="page">
    <label name="Label1" text="'state' Values ('SYN_BOOLEAN')"/>
    <check-box name="CheckBox1" attr-name="Login Disabled" attr-type="state" label="Account:"
text="disabled" radio-value="true"/>
    <label name="Label2" text=""/>
    <label name="Label3" text="'string' Values ('SYN_CI_STRING')"/>
    <check-box name="CheckBox2" attr-name="Given Name" attr-type="string" text="Alistair" radio-
value="Alistair" label="Given Name:" />
    <check-box name="CheckBox3" attr-name="Given Name" attr-type="string" text="James" radio-value="James"
label=""/>
    <check-box name="CheckBox4" attr-name="Given Name" attr-type="string" text="Sherlock" radio-
value="Sherlock" label=""/>
    <check-box name="CheckBox5" attr-name="Given Name" attr-type="string" text="Hercule" radio-
value="Hercule" label=""/>
    <label name="Label4" text=""/>
    <label name="Label5" text="'dn' Values ('SYN_DIST_NAME')"/>
    <check-box name="CheckBox6" attr-name="Group Membership" attr-type="dn" label="Groups:" text="Group1"
radio-value="Group1.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security Equals"/>
    <check-box name="CheckBox7" attr-name="Group Membership" attr-type="dn" label="Group2" radio-
value="Group2.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security Equals"/>
    <check-box name="CheckBox8" attr-name="Group Membership" attr-type="dn" label="Group3" radio-
value="Group3.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security Equals"/>
    <check-box name="CheckBox9" attr-name="Group Membership" attr-type="dn" label="Group4" radio-
value="Group4.Groups.NCS" sync-target="Member, Equivalent To Me" sync-source="Security Equals"/>
  </page-snapin>
</adv>
```

Screenshot of "CheckBoxes" in Action



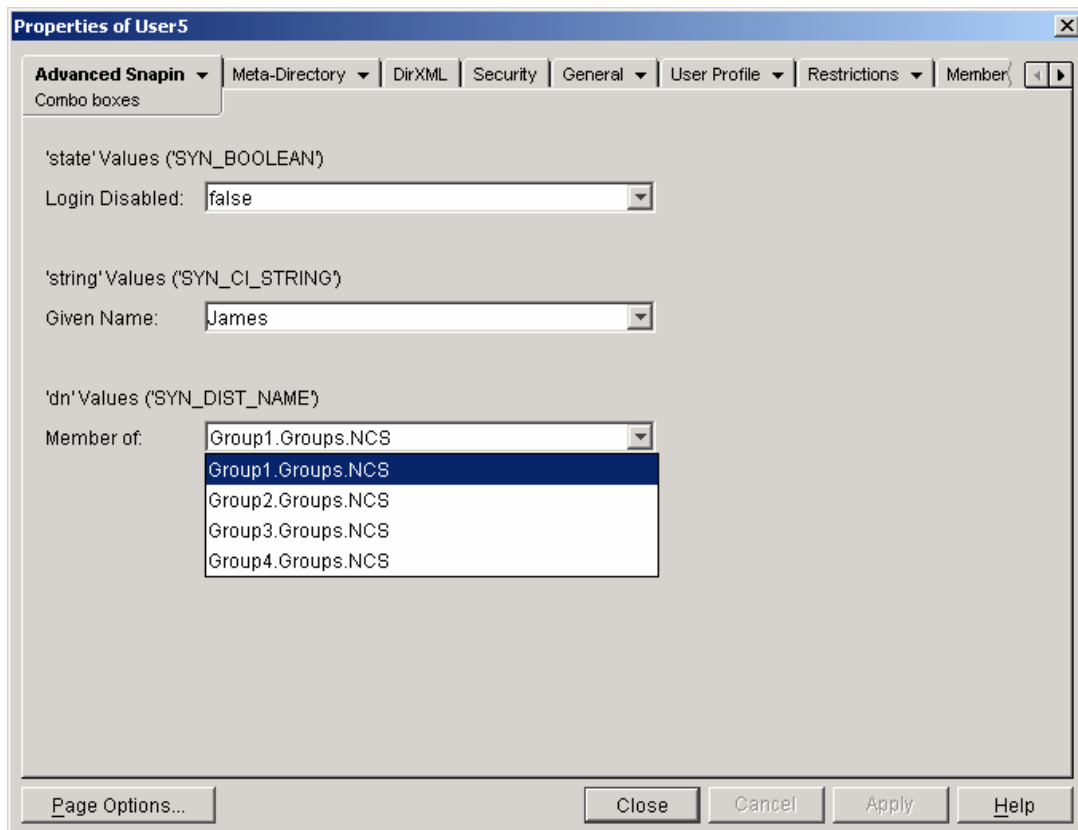
ComboBoxes

```

XML Code
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="ComboBoxes" snapin-description="Show functionality of combo boxes" snapin-tab-
name="Advanced Snapin" snapin-menu-name="Combo boxes">
    <label name="Label1" text="'state' Values ('SYN_BOOLEAN')"/>
    <combo-box name="combobox1" label="Login Disabled:" attr-name="Login Disabled" attr-type="state"
editable="false">
      <item value="false"/>
      <item value="true"/>
    </combo-box>
    <label name="Label2" text=""/>
    <label name="Label3" text="'string' Values ('SYN_CI_STRING')"/>
    <combo-box name="combobox2" label="Given Name:" attr-name="Given Name" attr-type="string"
editable="true">
      <item value="Alistair"/>
      <item value="James"/>
      <item value="Sherlock"/>
      <item value="Hercule"/>
    </combo-box>
    <label name="Label4" text=""/>
    <label name="Label5" text="'dn' Values ('SYN_DIST_NAME')"/>
    <combo-box name="combobox3" label="Member of:" attr-name="Group Membership" attr-type="dn" sync-
target="Member, Equivalent To Me" sync-source="Security Equals" enabled="true" editable="false">
      <item value="Group1.Groups.NCS"/>
      <item value="Group2.Groups.NCS"/>
      <item value="Group3.Groups.NCS"/>
      <item value="Group4.Groups.NCS"/>
    </combo-box>
  </page-snapin>
</adv>

```

Screenshot of "ComboBoxes" in Action



XML Panels

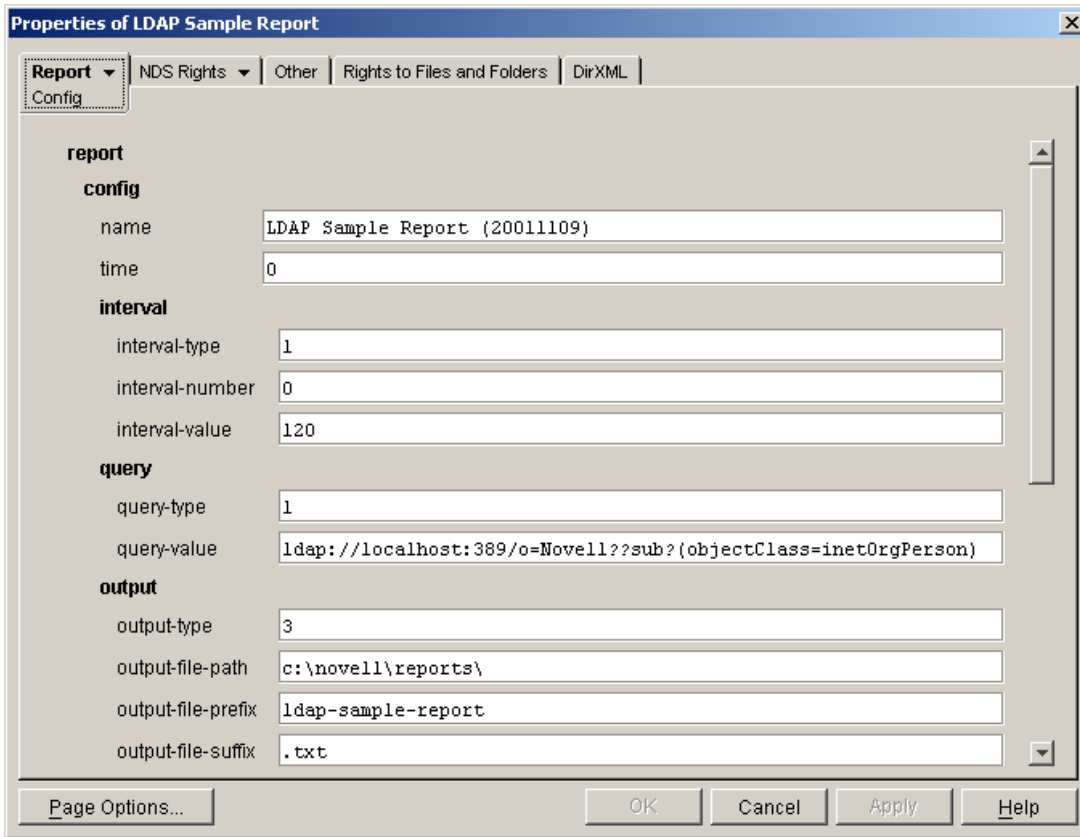
XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Volker Scheuber (private) -->
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="XML Editor" snapin-description="Allow editing of report configuration" snapin-
tab-name="Report" snapin-menu-name="Config" class="User" scope="page">
    <xml-panel name="XmlPanell" label="" attr-name="XmlData"/>
  </page-snapin>
</adv>
```

XML Code to be displayed with an <xml-panel> element

```
<report>
  <config>
    <name>LDAP Sample Report (20011109)</name>
    <time>0</time>
    <interval>
      <interval-type>1</interval-type>
      <interval-number>0</interval-number>
      <interval-value>120</interval-value>
    </interval>
    <query>
      <query-type>1</query-type>
      <query-value>ldap://localhost:389/o=Novell??sub?(objectClass=inetOrgPerson)</query-value>
    </query>
    <output>
      <output-type>3</output-type>
      <output-file-path>c:\novell\reports\</output-file-path>
      <output-file-prefix>ldap-sample-report</output-file-prefix>
      <output-file-suffix>.txt</output-file-suffix>
      <output-email-addr>vscheuber@novell.com</output-email-addr>
      <output-email-subj>Latest LDAP Sample Report</output-email-subj>
      <output-email-msg>Please find the latest LDAP Sample Report attached to this message.</output-email-
msg>
      <output-header>1</output-header>
      <output-footer>1</output-footer>
      <output-delimiter>;</output-delimiter>
      <output-fields>$$,$dn,"fix string value",givenName,sn,fullName,title,mail,ou</output-fields>
      <output-sort>sn</output-sort>
    </output>
    <notify>
      <notify-level/>
      <notify-email-addr>vscheuber@novell.com</notify-email-addr>
    </notify>
  </config>
</report>
```

Screenshot of "XML Panel" in Action



Rob Rawson's Plug-Ins in Action

The following snap-in uses quite a lot of Rob's plug-ins and demonstrates pretty well the power and flexibility of plug-ins. The following plug-ins are used:

- Convert Leaf to Root Most
- amHeritage
- Open Property Page
- parseMe
- parseMeToo
- Dynamic Combo Box

XML Code

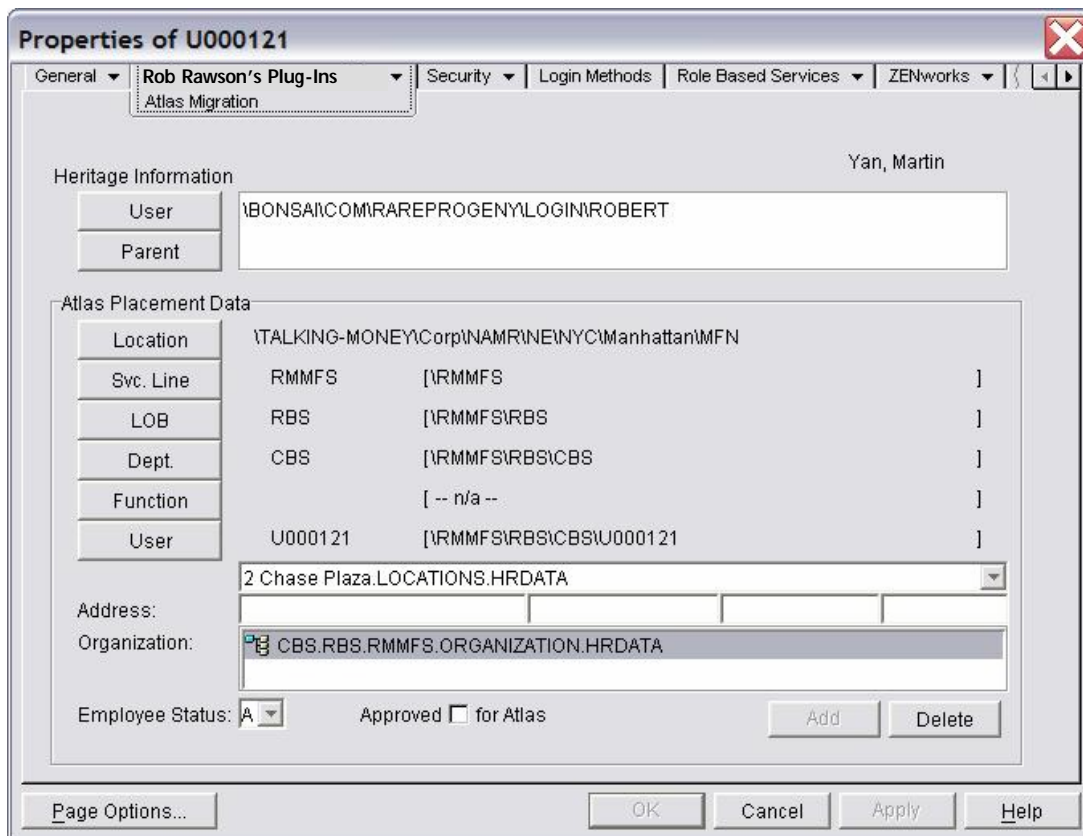
```
<?xml version="1.0" encoding="UTF-8"?>
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="jpmcMigration2" snapin-description="Atlas Migration" snapin-tab-name="J.P. Morgan
Chase and Co." snapin-menu-name="Atlas Migration" layout="absolute">
  <label constraints="130,110,484,20" name="Label508" text="Location Not Set"
action="setValue(getAttrValue(getAttrValue('jpmcRelationship4'),'jpmcString0'))"/>
  <label constraints="122,301,475,70" name="Label500" text=".ORGANIZATION.HRDATA"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.leaf2RootMost',getAttrValue('jpmcRelationship0'
))"/>
  <dn-panel constraints="120,300,480,75" filter="Organizational Unit" multi-valued="false" rows="1"
editable="true" name="dnField102" attr-name="jpmcRelationship0"/>
  <label constraints="240,235,340,20" name="LabelUIDFull" text="userName"
action="setValue(concat(getValue('Label500'),'U',getAttrValue('workforceID')))/>
  <label constraints="235,235,5,20" name="Level4Bracket1" text="["/>
  <label constraints="580,235,5,20" name="Level4Bracket2" text="]"/>
  <label constraints="140,235,100,20" name="UIDLabel" text="UID"
action="setValue(concat('U',getAttrValue('workforceID')))/>
  <button constraints="20,30,90,25" name="HeritageUserButton" text="User"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.amHeritage')"/>
  <button constraints="20,55,90,25" name="HeritageParentButton" text="Parent"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.amHeritage')"/>
  <button constraints="20,235,90,25" name="Button1" text="User"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',
concat(getValue('Label508'),getValue('LabelUIDFull')))/>
  <button constraints="20,110,90,25" name="Level0" text="Location"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',getValue('Label508'))"/>
  <button constraints="20,135,90,25" name="Level1" text="Svc. Line"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',concat(getValue('Label508'),ge
tValue('Level1Label')))/>
  <button constraints="20,160,90,25" name="Level2" text="LOB"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',concat(getValue('Label508'),ge
tValue('Level2Label')))/>
  <button constraints="20,185,90,25" name="Level3" text="Dept."
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',concat(getValue('Label508'),ge
tValue('Level3Label')))/>
  <button constraints="20,210,90,25" name="Level4" text="Function"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.openPropertyPage',concat(getValue('Label508'),ge
tValue('Level4Label')))/>
  <label constraints="500,1,200,20" name="whoIsIt" action="setValue(getAttrValue('Full Name'))"/>
  <label constraints="5,10,120,20" name="hTitle" text="Heritage Information"/>
  <!-- <border constraints="1,10,610,80" name="HeritageBorder" text="Heritage Information" /> -->
  <text-field constraints="120,30,480,50" name="HeritageText" attr-name="jpmcString1"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.amHeritage')" multi-valued="true"/>
  <!-- Location and Organization -->
  <label constraints="20,280,120,20" name="Label109" text="Address:"/>
  <label constraints="240,135,340,20" name="Level1Label" text="\1"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMe',concat('\1',getValue('Label500')))/>
  <label constraints="235,135,5,20" name="Level1Bracket1" text="["/>
  <label constraints="580,135,5,20" name="Level1Bracket2" text="]"/>
  <label constraints="140,135,100,20" name="Level1Label2" text="\\"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMeToo',getValue('Level1Label'))"/>
  <label constraints="240,160,340,20" name="Level2Label" text="\2"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMe',concat('\2',getValue('Label500')))/>
  <label constraints="235,160,5,20" name="Level2Bracket1" text="["/>
  <label constraints="580,160,5,20" name="Level2Bracket2" text="]"/>
  <label constraints="140,160,100,20" name="Level2Label2" text="\\"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMeToo',getValue('Level2Label'))"/>
  <label constraints="240,185,340,20" name="Level3Label" text="\3"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMe',concat('\3',getValue('Label500')))/>
  <label constraints="235,185,5,20" name="Level3Bracket1" text="["/>
  <label constraints="580,185,5,20" name="Level3Bracket2" text="]"/>
  <label constraints="140,185,100,20" name="Level3Label2" text="\\"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMeToo',getValue('Level3Label'))"/>
  <label constraints="240,210,340,20" name="Level4Label" text="\4"
```

```

action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMe',concat('\4',getValue('Label500'))) "/>
<label constraints="235,210,5,20" name="Level4Bracket1" text="["/>
<label constraints="580,210,5,20" name="Level4Bracket2" text="]"/>
<label constraints="140,210,100,20" name="Level4Label2" text="\
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.parseMeToo',getValue('Level4Label')) "/>
<combo-box constraints="120,260,480,20" name="combo999" label="Location:" attr-type="dn"
editable="false" attr-name="jpmcRelationship4" rows="1"
action="callPlugin('com.novell.ncs.consoleone.advanced.plugins.MyNextPlugin','LOCATIONS.HRDATA') ">
  <item value=""/>
</combo-box>
<text-field constraints="300,280,120,20" name="Text501"
action="setValue(getAttrValue(getValue('combo999','true'),'Physical Delivery Office Name'))" attr-
name="Physical Delivery Office Name"/>
<text-field constraints="420,280,100,20" name="Text502"
action="setValue(getAttrValue(getValue('combo999','true'),'S'))" attr-name="S"/>
<text-field constraints="520,280,80,20" name="Text503"
action="setValue(getAttrValue(getValue('combo999','true'),'Postal Code'))" attr-name="Postal Code"/>
<text-field constraints="120,280,180,20" name="Text504"
action="setValue(getAttrValue(getValue('combo999','true'),'SA'))" attr-name="SA"/>
<label constraints="20,300,120,20" name="Label02" text="Organization:"/>
<label constraints="196,345,75,20" name="Label03" text="Approved"/>
<checkbox constraints="250,345,120,20" name="CheckBox1" attr-name="jpmcFlag0" label="Approved"
text="for Atlas" radio-value="true"/>
<label constraints="20,345,120,20" name="Label02" text="Employee Status:"/>
<combo-box constraints="120,345,30,20" name="Combo3" attr-name="jpmcStatusCode">
  <item value="A"/>
  <item value="L"/>
  <item value="P"/>
  <item value="T"/>
</combo-box>
<border constraints="1,90,610,300" name="Border2" text="Atlas Placement Data"/>
</page-snapin>
</adv>

```

Screenshot



The next example shows:

- Selection List Control

XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<adv dtd="adv.dtd" version="1.0">
  <page-snapin snapin-name="jpmcMigration2" snapin-description="Atlas Migration" snapin-tab-name="J.P. Morgan
Chase and Co." snapin-menu-name="Virtual Campus" layout="absolute">
    <label constraints="20,0,240,20" name="Label101" text="Associated Buildings"/>
    <label constraints="360,0,240,20" name="Label101" text="Other Buildings"/>
    <button constraints="280,80,60,20" name="addButton" text="&lt;&lt;"/>
    <button constraints="280,120,60,20" name="removeButton" text="&gt;&gt;"/>
    <text-field constraints="120,0,160,20" attr-name="jpmcBuildings" name="Text201"
    multi-valued="true"/>
  </page-snapin>
</adv>
```

Screenshot

