

---

# Understanding Password Reset

---

## Password Reset

---



**Note:** This article applies to Fuji and earlier releases. For more current information, see *Password Reset and Password Change Applications* <sup>[1]</sup> at <http://docs.servicenow.com> **The ServiceNow Wiki is no longer being updated. Visit <http://docs.servicenow.com> for the latest product documentation.**

### Overview

The Password Reset application helps organizations implement and monitor a customizable self-service or service-desk process for resetting and changing passwords on the local ServiceNow instance. Subscribing to the Password Reset Orchestration Add-on provides the ability to reset passwords on Active Directory and other credential stores. To see an introductory video on Password Reset, go to [Introducing Password Reset \(Video\)](#) <sup>[2]</sup>.

Password reset is available starting with the Dublin release.

### Password Reset Process Flow

The password reset life cycle consists of:

- **Planning** to ensure that all applicable organizational guidelines, security policies, and areas of the organization are considered. [Click here to view planning procedures.](#)
- **Setting up** the password reset process according to the plan. [Click here to view setup procedures.](#)
- **Resetting** passwords. [Click here to view reset procedures.](#)
- **Monitoring** password reset activity to identify security threats and to ensure compliance with the organization's password policy requirements. [Click here to view monitoring tasks.](#)

To see a video on setting up Password Reset, go to [Setting Up a Service Desk-Assisted Password Reset Process \(Video\)](#) <sup>[3]</sup>.

### Planning for Password Reset

While planning your password reset implementation consider:

- **Groups:** Analyze and assess how members of each group in your organization access the system. For example, if members of the sales group primarily access the system remotely, consider using a stronger method or multiple methods for verifying each user's identity.
  - **Roles:** Identify user roles that have access to critical information and resources. For example, stronger verification methods may be required for roles that have access to employee data, accounting information, or network configurations.
  - **Verification methods:** Determine the number and variety of verification methods needed for the different password reset processes based on the analysis of groups and roles.
  - **Credential store:** Determine if single sign-on is enabled with the type of directory service or other credential store used. If the directory service is configured for single sign-on, consider increasing the level of security by using multiple methods to verify a user's identity. A compromised user name and password can easily allow access to associated systems in a single sign-on environment.
-

- **Enrollment:** Consider how enrollment will be implemented within the organization. For example, will enrollment in the password reset program be optional or required? How will users be notified to enroll in the program? Will users be auto-enrolled in the program? The answers to these questions will help you determine the appropriate verification types to use.
- **Password reset process:** Consider the password reset options the organization wants to offer users. Will users reset their own passwords from a self-service module or will the service-desk reset passwords on behalf of users? If the organization uses single sign-on, how will users reset their password if they are unable to log on? What options are available to users working offsite?

To make the Password Reset application with Orchestration available to all users publicly, create a new Password Reset Process only for this purpose and make it accessible to Public. Create and publish a new URL for the process.

## Setting Up Password Reset

An administrator or user with the `password_reset_admin` role configures password reset with the following elements:

- **Verifications:** define methods for confirming a user's identity during enrollment and during a password reset. [Click here for instructions on configuring verifications.](#)
- **Credential Stores:** specify where to find login credentials such as user names and passwords. [Click here for instructions on configuring credential stores.](#)
- **Processes:** define how users reset their passwords. [Click here for instructions on defining the password reset processes.](#) To configure the password reset process, you specify:
  - The verification methods used to authorize the password reset.
  - The credential store containing the user's login credentials.
  - The user groups authorized to use password reset.

You can set up password reset to match your organization's preferred access method:

- **Self-service:** users reset their password over the Internet from a publicly accessible web page. Administrators must publish the URL to the password reset form.
- **Service desk:** users reset their passwords with the assistance of a service-desk employee over the phone or in person. Service-desk employees must have the `password_reset_service_desk` role to perform password resets on behalf of users.

To see a video of the user's perspective on resetting a password, go to [Resetting User Passwords \(Video\)](#) <sup>[4]</sup>.

## Monitoring Password Reset Activity

The Password Reset application provides a variety of tools to help you monitor password reset activity for potential security threats, compliance with organizational password policies, and any problems with password reset processes and verifications. For more information, see [Monitoring Password Reset Activity](#).

## Password Change

Password change extends the Password Reset application by letting administrators define how users change their passwords from a self-service module.

Password change is available starting with the Fuji release.

## Password Change Process

The password change process consists of the following steps:

1. The user logs in to the instance.
2. The user selects the **Change Password** module or link from the user profile record.
3. The user selects the credential store where their password resides.
4. The user provides the old and new passwords.
5. The appropriate workflows validate the old and new passwords.

## Setting Up Password Change

An administrator or user with the `password_reset_admin` role can configure the password change process with the following records:

- **Credential Stores:** specify where to find login credentials such as user names and passwords. Click here for instructions on configuring credential stores.
- **Password Change Process:** define the process that users follow to change their passwords. Click here for instructions on defining the password change process. To configure the password change process, you specify:
  - The credential store containing the user's login credentials.
  - The user groups authorized to change their passwords.

The password change application supports only a self-service password change process. An administrator must publish the URL to the password change form.

## Monitoring Password Change Activity

Administrators can monitor password change activity for potential security threats, ensure compliance with organizational password policies, and identify problems with password change. See [Monitoring Password Reset Activity](#).

## Mobile Support

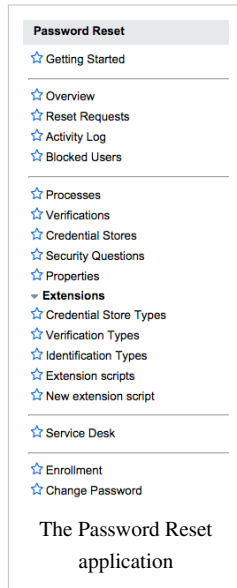
Administrators can enable support for password change from tablets and mobile devices.

## Roles

Role title [name]	Description
password reset administrator [password_reset_admin]	Sets up the processes for password reset and password change.
service desk agent [password_reset_service_desk]	Resets passwords on behalf of users, tracks password reset requests, and views logs.
credentials manager [password_reset_credential_manager]	Determines which credential stores are valid for use with password reset.

## Menus and Modules

To access password reset and change features, use these modules under the Password Reset application.



- **Overview:** Access a portal with charts and data that show activities like password reset requests, attempts, and enrollment status.
- **Reset Requests:** View password reset and password change requests.
- **Activity Log:** View the log of password reset and password change activity.
- **Blocked Users:** View all users who are currently blocked from password reset and password change. (In the Dublin release, blocked users are referred to as *locked* users with "Locked=true" on the Password Reset User Lockouts list.)
- **Processes:** Define and configure the password reset or password change process for users.
- **Verifications:** Define methods to verify a user's identity during password reset.
- **Credential Stores:** Provides access to user information maintained in stores such as ServiceNow User [sys\_user] tables and Active Directory servers.
- **Security Questions:** Allows adding security questions or modifying existing security questions.
- **Properties:** Displays a list of all properties and settings for password reset and password change.
- **Extensions:** Modify and create features that extend password reset functionality: Credential store, Verification, and Identification. Provides access to scripts. For more information on scripting in password reset, see Creating Extension Scripts.
- **Extension scripts:** View, modify, and create scripts to extend the functionality of the application.
- **New Extension script:** Create new extension scripts from a set of predefined script types.
- **Service Desk:** [admin only] Reset a user password.
- **Enrollment:** Allows users with one or more applied processes to enroll in password reset.
- **Change Password:** Allows users to change their passwords.

## Activating Password Reset and Password Change

You must activate the Password Reset plugin to access password reset or password change features.

Click the plus to expand instructions for activating a plugin.

If you have the admin role, use the following steps to activate the plugin.

1. Navigate to **System Definition > Plugins**.
2. Right-click the plugin name on the list and select **Activate/Upgrade**.

If the plugin depends on other plugins, these plugins are listed along with their activation status.

3. [Optional] If available, select the **Load demo data** check box.

Some plugins include demo data—sample records that are designed to illustrate plugin features for common use cases. Loading demo data is a good policy when you first activate the plugin on a development or test instance. You can load demo data after the plugin is activated by repeating this process and selecting the check box.

4. Click **Activate**.

## Activating the Password Reset - Orchestration Add-on

For additional functionality, including the ability to reset passwords on Active Directory, you must use the Password Reset - Orchestration Add-on, which provides two credential store types that are not available in the basic Password Reset application. The Orchestration Add-on is available as a separate subscription. For more information, contact your ServiceNow account representative.

## Enhancements

### Fuji

- Supports changing passwords in addition to resetting passwords.
- Provides a redesigned user interface for resetting passwords on Internet Explorer 10 and later.
- Provides a Windows add-on to support resetting passwords on a locked PC. This feature is also supported on Dublin and Eureka.

### Eureka

- A locked user account can now be unlocked without resetting the user's password.
- The user's lock state is now displayed in the password reset form.
- A report shows the number of password reset requests by action type—**Reset Password, Unlock Account, or Reset and Unlock**.
- The `password_reset.request.unlock_window` property is available to set how long a user must wait after a successful unlock operation before starting a new request.
- Workflows for unlocking user accounts on a credential store are available.

## References

- [1] [https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c\\_PasswordReset.html](https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c_PasswordReset.html)
  - [2] <http://www.youtube.com/watch?v=IakvJpNGpZ4I>
  - [3] <http://www.youtube.com/watch?v=wDxK2eachywI>
  - [4] <http://www.youtube.com/watch?v=CokNL8aw6hA>
-

---

# Configuration

---

## Configuring Verifications

---

### Overview

The Password Reset application uses verifications to confirm the identity of users who are attempting to reset their password. The application comes with several preconfigured verifications. Password reset administrators can use them or create new verifications for use with a password reset process.

### Installed Verifications

Password reset comes with four verifications that you can use as configured. Installing the demo data adds two sample verifications that you can use as a starting point for creating custom verifications:

- **SMS Verification:** used with a self-service or service-desk password reset model that relies on auto-generated code numbers.
- **QA Verification:** used with a self-service password reset model that relies on questions and answers.
- **Personal Data - Enter User Name:** used with a self-service password reset model that relies on information available in the system.
- **Personal Data - Confirm Email Address:** used with a self-service password reset model that relies on information available in the system.
- **Sample Mock Verification #1:** used to create custom verifications.
- **Sample Mock Verification #2:** used to create custom verifications.

### Creating Password Reset Verifications

All verifications are based on specific verification types, which are like templates that provide a desired set of capabilities.

To create a verification:

1. Navigate to **Password Reset > Verifications**.
  2. Click **New**.
  3. Fill in the fields, as appropriate. (See table.)
  4. Click **Submit**.
-

Password Reset Verification

Required field

Submit

Name:

Description:

Type:

Order:

100

Password Reset Verification Parameters

Name	Value	Order	Description
Insert a new row...			

Submit

New verification form

Field	Description
Name	Unique and descriptive name of the new verification.
Description	Description of the new verification.
Type	Verification type that that provides the desired capabilities. Verification methods inherit the functionality of the selected type. You can use the default verification types, listed below, modify them to meet your needs, or create new ones. <ul style="list-style-type: none"><li>Personal Data Confirmation Verification</li><li>Personal Data Verification</li><li>Security Question Verification</li><li>SMS Code Verification</li></ul>
Order	Position of the verification as it appears on the Enrollment and Password Reset forms.
Password Reset Verification Parameters	Parameters used by a verification to configure specific behaviors, like number of questions required to enroll, request expiration time, and columns used. Set parameters for any behavior that should be different from the default specified in the Password Reset Properties. <p>The available parameters are described separately for each verification type.</p>

### SMS Code Type Verifications

Simple Message Service (SMS) code verifications allow users to verify their identity with the help of an SMS enabled device, such as a cell phone that accepts text messages. When a user requests a password reset, the password reset application sends a code to the SMS device that is registered to the user. The code is a number that the user must enter to verify their identity. You can change the default behavior of an SMS code verification by setting parameter values, as shown below.

Password Reset Verification

Required field

Update

Delete

Name:

SMS Verification

Description:

Type:

SMS Code Verification Ty

Order:

100

Password Reset Verification Parameters

Name	Value	Order	Description
complexity	4	100	The number of digits in the verification...
expiry	4	200	The number of minutes the verification c...
pause_window	2	300	
max_per_day	10	400	
Insert a new row...			

Update

Delete

Example SMS verification parameters

Parameter Name	Description
<b>expiry</b>	Number of minutes the verification code is valid. <b>Data Type:</b> Integer (any positive integer) <b>Default Value:</b> 5
<b>complexity</b>	Number of digits in the verification code sent to user. <b>Data Type:</b> Integer (any positive integer) <b>Default Value:</b> 4
<b>pause_window</b>	Number of minutes before the user can attempt to send another SMS code for verification. <b>Data Type:</b> Integer (any positive integer) <b>Default Value:</b> 2
<b>max_per_day</b>	Maximum number of SMS codes sent for verification per day. <b>Data Type:</b> Integer (any positive integer) <b>Default Value:</b> 10

Create only one instance of each SMS code parameter. Attempting to create additional parameters causes an error.

## Security Question Type Verifications

Security question verifications require that users answer questions that only they know. During the enrollment process a user selects and answers a set of questions. Those questions and answers are saved for when the user requests a password reset. The QA Verification is based on the Security Question Verification Type. You can change the default behavior of a security question verification by setting parameter values, as shown below.

Example security question verification parameters

Parameter Name	Description
<b>num_enroll</b>	Number of security questions displayed during enrollment. <b>Data Type:</b> Integer (any positive integer that does not exceed the number of questions in the security questions list) <b>Default Value:</b> 5
<b>num_reset</b>	Number of security questions displayed during the password reset request. <b>Data Type:</b> Integer (any positive integer that does not exceed the value of num_enroll) <b>Default Value:</b> 3

Create only one instance of each security question parameter. Attempting to create additional parameters causes an error.



## Adding Security Questions

To add a new security question:

1. Navigate to **Password Reset > Security Questions**.
2. Click **New**.
3. Enter the new question in the **Security Question** field.
4. Click **Submit**.

## Personal Data and Personal Data Confirmation Type Verifications

Personal data verifications allow users to verify their identity by providing answers to questions that are generated from personal information stored in the User [sys\_user] table. Users are typically not required to go through an enrollment step if they are associated with a password reset process that uses a personal data verification.

Personal data confirmation verifications allow employees with the service-desk role to access personal data from the sys\_user table when assisting a user with a password reset request.

For personal data and personal data confirmation verifications, parameters are specified as name/value pairs that correspond to a particular piece of user information. The example verifies users by their email address: the label parameter value (the text that the user sees) is **Email address** and the column parameter value (the column\_name from the sys\_user table that must match) is **email**.

- Password reset supports only one set of name/value pair parameters per verification. For example, the num\_enroll and num\_reset properties specify the number of questions to ask during the enrollment and verification processes. Verification parameters that you define here take precedence over the password property settings.
- Additional parameters are ignored. To use multiple pieces of personal information for user verification, create additional personal data or personal data confirmation verifications and add those to the related password reset process.

The screenshot shows a web interface for configuring a 'Password Reset Verification'. The title bar indicates it is a 'Required field' and includes 'Update' and 'Delete' buttons. The 'Name' field is set to 'Personal Data - Confirm Email Address'. The 'Type' is 'Personal Data Confirmation' and the 'Order' is 400. Below this is a table titled 'Password Reset Verification Parameters' with columns for Name, Value, Order, and Description. The table contains two rows: one with 'label' as the name and 'Email address' as the value (Order 100), and another with 'column' as the name and 'email' as the value (Order 200). There are 'Update' and 'Delete' buttons at the bottom of the table.

Name	Value	Order	Description
label	Email address	100	
column	email	200	

Example personal data verification parameters

Parameter Name	Description
<b>label</b>	Text that is displayed to the user during the password reset request. <b>Data Type:</b> String <b>Default Value:</b> n/a
<b>column</b>	The column_name from the sys_user table that provides the data to verify the user's identity. <b>Data Type:</b> String <b>Default Value:</b> n/a

## Deleting Verifications

Before deleting a verification, make sure that it is not being used by a password reset process. If a verification is being used by a process, remove it from the process first.

To delete a verification:

1. Navigate to **Password Reset > Verifications**.
2. Select the check box next to the verification to delete.
3. In the **Actions** choice list below the list, select **Delete**.

## Configuring Credential Stores

---



**Note:** This article applies to Fuji. For more current information, see *Credential Stores for Password Reset*<sup>[1]</sup> at <http://docs.servicenow.com>. The Wiki page is no longer being updated. Please refer to <http://docs.servicenow.com> for the latest product documentation.

## Overview

Credential stores such as the ServiceNow User [sys\_user] table store user information such as user names and passwords that can be used as login credentials. Users with the password\_reset\_admin or password\_reset\_credential\_manager role can create and modify connections to credential stores.

If you have subscribed to the Password Reset Orchestration Add-on and need to connect to an Active Directory credential store or another directory-like service that relies on the SOAP protocol, complete the procedures listed on the Configuring Remote Credential Stores page before beginning the procedures on this page.

# Creating Credential Stores

- 1. Navigate to **Password Reset > Credential Stores**.
- 2. Click **New**.
- 3. Fill in the fields, as appropriate. (See table.)
- 4. Click **Submit**.

Password

Required field

Submit

Save & Test Connection

Name:

Description:

Type:

Auto generate password:

-- None --

Hostname:

User account lookup:

-- None --

Password rule hint:

Path: p

Password rule:

1

// This script is invoked during the password reset step and is used to

2

// ensure that the new password is in compliance with password policy.

3

// By default this script always returns true.

4

5

function isValidPassword(password) {

6

// TODO: add validation logic here

7

return true;

8

}

New credential store form

Field	Description
Name	Unique name of the new credential store.
Description	Description of the credential store.
Type	<p>Credential store type, for example, a local ServiceNow instance. Credential store types are like templates that provide a desired set of capabilities. Credential stores inherit the functionality of the credential store type.</p> <p>Installed credential store types:</p> <ul style="list-style-type: none"><li>• <b>Local ServiceNow Instance:</b> installed with Password Reset.</li><li>• <b>AD Credential Store:</b> installed with the Orchestration Add-on.</li><li>• <b>Remote (SOAP) ServiceNow:</b> installed with the Orchestration Add-on.</li></ul> <p>For more information, see Defining Credential Store Types.</p>
Auto generate password	[Optional] Script include that generates a new password automatically.
Hostname	URL or IP address of the credential store that contains the user names and passwords. You must use a the URL hostname, not an IP address, when the type is <b>AD credential store</b> .
User account lookup	[Optional] Script include that maps the user's ServiceNow ID to the user's credential store ID. A default script, PwdDefaultUserAccountLookup, is provided that returns the user's ServiceNow user name.
Password rule hint	[Optional] Password complexity rules that the user sees during a password reset.
Password rule	[Optional] Client script that validates the password the user enters. The script is invoked after the user enters a new password and clicks the <b>Password Reset</b> button. This script can be used to provide additional strength to the password.

## Testing Connections to Credential Stores

Test a connection to a credential store when creating a new credential store, or when users are experiencing problems that you suspect are due to connection issues related to a credential store. A connection test workflow is needed to test a connection.

To test a connection to a credential store:

1. Navigate to **Password Reset > Credential Stores**.
2. Open a credential store.
3. In the header bar, click the **Save & Test Connection** button.

A progress page appears and displays the results of the test.

## Deleting Credential Stores

Before deleting a credential store, check all password reset processes to ensure that the credential store is not in use. If it is being used by a process, remove it.

To delete a credential store:

1. Navigate to **Password Reset > Credential Stores**.
2. Select the check box next to the credential store to delete.
3. In the action choice list below the list, select **Delete**.

## References

[1] [https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c\\_CredentialStores.html](https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c_CredentialStores.html)

## Configuring Processes

---



**Note:** This article applies to Fuji. For more current information, see *Password Reset and Password Change Applications* <sup>[1]</sup> at <http://docs.servicenow.com>. The ServiceNow Wiki is no longer being updated. Please refer to <http://docs.servicenow.com> for the latest product documentation.

## Overview

Configure processes to specify how users reset or change their password. You must have either the administrator or password\_reset\_admin user roles to create a password reset or change process.

## Creating Password Reset Processes

A *password reset* process consists of the following elements:

- The verification methods used to authorize the password reset.
- The credential store containing the user's login credentials.
- The user groups authorized to use password reset.



**Note:** If your instance is using an LDAP integration and the Active Directory settings require users to reset their password upon login, your users will not be able to log in the instance. The instance cannot change any user's active directory password.

To create a new password reset process:

1. Create the credential store record for user names and passwords to be managed.
2. Navigate to **Password Reset > Processes**.
3. Click **New**.
4. Fill in the fields (see Password Reset Process Form Fields).
  - Select the **Password reset** option.
  - Select the **Credential store** containing the user credentials to which this process applies.
  - Select the **Apply to all users** option if applicable.
5. [Optional] Select advanced options (see Advanced Configuration Options).
6. Right-click the header bar and select **Save**.
7. From the **Verifications** related list, select one or more verification methods.
8. [Optional] From the **Groups** related list, select one or more user groups that can reset their passwords.
9. Click **Update**.

The screenshot shows the 'Password Reset Process - Demo Self-Service Process 1' configuration page. The form includes the following sections:

- Basic Information:** Name (Demo Self-Service Process 1), Description (Simple process for demoing various reports).
- Credential Store:** Local ServiceNow Instance (selected), Active (checked).
- Options:** Password change (unchecked), Password reset (checked), Apply to all users (unchecked).
- Password Reset Details:**
  - Public access (checked), Public URL (/spwd\_reset.do?sysparm\_url=demo1), URL suffix (demo1), Display captcha (checked), Identification type (Email Identification).
  - Enable account unlock (unchecked), Auto-generate password (unchecked).
- Advanced:** Entry UI macro, Success UI macro, Failure UI macro, Post reset script (PwdPostProcessor).

Reset Process Form Fields).

- Select the **Password change** option.
- Select the **Credential store** containing the user credentials to which this process applies.

## Creating Password Change Processes

A *password change* process consists of the following elements:

- The credential store containing the user's login credentials.
- The user groups authorized to use password change.

To create a new password change process:

1. Create the credential store record for user names and passwords to be managed.
2. Navigate to **Password Reset > Processes**.
3. Click **New**.
4. Fill in the fields (see Password

- Select the **Apply to all users** option if applicable.
5. [Optional] Select advanced options (see Advanced Configuration Options).
  6. Right-click the header bar and select **Save**.
  7. [Optional] From the **Groups** related list, select one or more user groups that can change their passwords.
  8. Click **Update**.

## Password Reset Process Form Fields

The Password Reset Process form has the following fields.

Field	Description
Name	Unique name for the process.
Description	Description of the process.
Credential store	Credential store containing login credentials.
Password change	Check box for indicating that this process applies to a password change. This option is available only when <b>Password Reset</b> is selected. This check box is available starting with the Fuji release.
Password reset	Check box for indicating that this process applies to a password reset. The <b>Enable account</b> lock option becomes visible when you select this check box. This field is available starting with the Fuji release.
Active	Check box for enabling the process. This check box is available only after the record has been saved.
Apply to all users	Check box for assigning the process including enrollment requirements to all users in the system, regardless of group affiliation. Clearing the check box reassigns the process back to the groups listed in the <b>Groups</b> related list.
Public access	Check box for allowing public access to the password reset form through a URL. If this check box is cleared, only service desk employees can reset a password using this process.  This check box is available only when <b>Password reset</b> is selected.
Display captcha	Check box for displaying a CAPTCHA on the password reset page.  This check box is available only when <b>Public access</b> is selected.
URL suffix	Suffix used to create a unique URL for the password reset form.  This check box is available only when <b>Public access</b> is selected.
Public URL	URL of the page where users go to reset their password. The value from the URL suffix field is automatically appended to the URL when you tab out of the URL suffix field.  This check box is available only when <b>Public access</b> is selected.
Identification type	Allows you to create different identification methods for public password reset processes. These methods override the verification methods associated with the process. Users can enter different verification information like employee ID, email, and user ID to verify their identity. The Email and Username Identification types come with the Password Reset application. For more information on creating custom identification types, see Defining Identification Types. Some knowledge of JavaScript is recommended.
Enable account unlock	Check the box to enable each end user to unlock the user's account on the credential store without resetting the password.  If the check box is selected, then a button labeled <b>Unlock Account</b> appears on the end-user Change Password page. When a user clicks the button, the account is unlocked, but the password is not updated. (Eureka release)  This check box is available only when <b>Password reset</b> is selected.
Unlock user account	Check box for unlocking users' accounts on credential stores after a password reset. (Dublin release)
Auto generate password	Check box for automatically generating a new password for the user. When this check box is selected, you are required to select the <b>Email password</b> or <b>Display password</b> check box, or both. This is useful for service-desk-assisted processes.  This check box is available only when <b>Password reset</b> is selected.

**User must reset password** Check box for requiring users to reset their password immediately after logging in with the auto-generated password.



**Note:** *Users whose credentials are held in the local ServiceNow instance credential store are prompted to change their password the first time that they log in. Users whose credentials are held in an Active Directory credential store are not prompted to change their passwords in the ServiceNow instance. Such users must change their passwords from a computer on the domain.*

This check box is available only when **Auto generate password** is selected.

**Email password** Check box for emailing the new password to the user. While this is useful in a self-service process, it can also be used in a service-desk process. Depending on the password reset process that your organization uses, this option can provide another layer of security by requiring that users access their email to view the password. In a service-desk process, emailing the password to users ensures that only the user requesting the password reset can view the password.

This check box is available only when **Auto generate password** is selected.

**Display password** Check box for displaying the new password on the screen. In a self-service process, the password appears on the user's screen. In a service-desk process, the password appears on the service-desk employee's screen.

This check box is available only when **Auto generate password** is selected.

#### Related Lists

**Verifications** One or more verification methods that are used in the process.

This related list is available only after the record has been saved.

**Groups** ServiceNow user groups to associate with the process.

This related list is available only after the record has been saved and if **Apply to all users** is clear.

## Advanced Configuration Options

Advanced configuration allows you to select UI macros and script includes that extend the basic functionality of a process.

Field	Description
Entry UI macro	UI macro that displays a customized message to users when they access the initial password reset screen.
Success UI macro	UI macro that displays a customized message to users on the final password reset screen if their password was successfully reset.
Failure UI macro	UI macro that displays a customized message to users on the final password reset screen if their password reset failed.
Post reset script	Script include that performs actions after the password reset process completes whether the outcome is success or failure. For more information on customizing post processor scripts, see the <b>Post reset script</b> category described in Creating Extension Scripts.

## References

- [1] [https://docs.servicenow.com/bundle/jakarta-it-operations-management/page/administer/login/concept/c\\_PasswordReset.html](https://docs.servicenow.com/bundle/jakarta-it-operations-management/page/administer/login/concept/c_PasswordReset.html)

# Remote Credential Stores

---

## Overview

Remote credential stores, such as Active Directory, manage user names and passwords outside of the local ServiceNow instance. A remote credential store can also be a remote ServiceNow instance, a UNIX or Linux server, or any other directory-like service. The Password Reset Orchestration Add-on is required to be able to connect to remote credential stores. For password reset, the term remote credential store refers to any credential store other than the local ServiceNow instance.

## Configuring Password Reset and Active Directory

When the Orchestration Add-on is active, password reset can change passwords on an Active Directory credential store by referencing an Active Directory user role with the appropriate password change privileges.

Active Directory must have a user role with the following privileges:

- Descendent User objects
  - Reset password
  - Read/Write pwdlastset
  - Read/Write UserAccountcontrol
  - Write Account Restrictions
  - Read/Write lockouttime
  - Read MemberOf
- Descendent Group objects
  - Read Members
  - Read MemberOf

## Configuring the Connection to the Credential Store

1. Install a MID server on a Windows computer that can connect to Active Directory (AD).
2. Configure the MID server.
3. On the ServiceNow instance, navigate to **Orchestration > Credentials** and then click **New**.
4. On the **Credentials** page, select **Windows Credentials**.
  - For **User name**, enter your AD domain user. For example, `domain\admin`.
  - For **Password**, enter your AD domain user password.
  - For **Applies to**, you specify the MID server or servers that will access the AD server. You can specify **All MID servers** or **Specific MID Servers**. If you select **Specific MID Servers**, then select the servers from the list.
5. Create an Active Directory credential store.



## Setting Up SOAP Credentials for Password Reset

When the Orchestration Add-on is active, ServiceNow can use the SOAP protocol to interact with remote credential stores such as a remote ServiceNow instance.

1. Navigate to **System Web Services > SOAP Message**.
2. Click **Password Reset Request**.
3. From the **Soap Message Functions** related list, configure both the `password_reset` and `sys_user_get_record` functions:
  - In **Basic auth user ID**, enter the user ID for the remote system user who has privileges to update records on the ServiceNow User [sys\_user] table.
  - In **Basic auth user password**, enter the password for the remote system user who has privileges to update records on the ServiceNow User [sys\_user] table.
  - Select **Use basic auth**.
  - Click **Update**.



**Note:** You do not need to enter a value in the **SOAP endpoint** field. The field automatically shows the name of the ServiceNow instance used for password reset.

## Password Reset Desktop Integration (Ctrl+Alt+Del)

---

The Password Reset Windows Application enables a user who is locked out of a Windows computer to reset the password directly from the Windows login screen. Please review the following ServiceNow product documentation pages for more information:

- Admins: Password Reset Windows Application <sup>[1]</sup>
- Users: Reset your password on Windows systems <sup>[2]</sup>

### Determining which installer to use

The ServiceNow Password Reset Windows Application installer is available in two formats, both included in the installer .zip file:

- The .exe installation is best suited to testing or single-install scenarios
- You can use the .msi installation to distribute to a group of computers using a configuration management tool such as Microsoft's SCCM product

## References

[1] [https://docs.servicenow.com/bundle/geneva-it-operations-management/page/administer/login/concept/c\\_PasswordResetDesktopIntegration.html](https://docs.servicenow.com/bundle/geneva-it-operations-management/page/administer/login/concept/c_PasswordResetDesktopIntegration.html)

[2] [https://docs.servicenow.com/bundle/geneva-it-operations-management/page/administer/login/task/t\\_RsttPassDeskRst.html](https://docs.servicenow.com/bundle/geneva-it-operations-management/page/administer/login/task/t_RsttPassDeskRst.html)

---

# Using Password Reset

## Enrolling



**Note:** This article applies to Fuji and earlier releases. For more current information, see *Enroll in the Password Reset Program* <sup>[1]</sup> at <http://docs.servicenow.com> **The ServiceNow Wiki is no longer being updated. Visit <http://docs.servicenow.com> for the latest product documentation.**

## Overview

Users who have not been auto-enrolled in a password reset program must go through an enrollment procedure. The procedure may vary depending on the settings specified by the password reset administrator. You may need to enroll using security questions, SMS, or both.

## Security Question Enrollment

For security question enrollment, select a question from the list and enter the answer in the field below. Repeat the process until all question and answer fields are filled in.

1. Navigate to **Password Reset > Enrollment**.
2. Fill in the fields, as required.  
If needed, enroll one or more SMS enabled devices.
3. Click **Submit**.

**Enrollment for QA Verification**  
Complete the question and answer section below

Question 1 Select a question  
Answer

Question 2 Select a question  
Answer

Question 3 Select a question  
Answer

**Submit**

Example of a security question enrollment form

## SMS Code Enrollment

For SMS enrollment, you must enroll at least one verified SMS enabled device. If you previously added an SMS enabled device to your user profile, the name of the device automatically appears under **Available Devices** on the form. If your profile does not have a device, you can add one directly from the enrollment form.

To add a device using the enrollment form:

1. Enter the device name and phone number.
2. Select a service provider.
3. Click **Add Device**.

The device name appears under **Available Devices**.

To verify a device using the enrollment form:

1. Click **Verify** to have a code sent to the device.

2. Retrieve the code from the device and enter it in the **Enter code** field.
3. Click **Ok**.

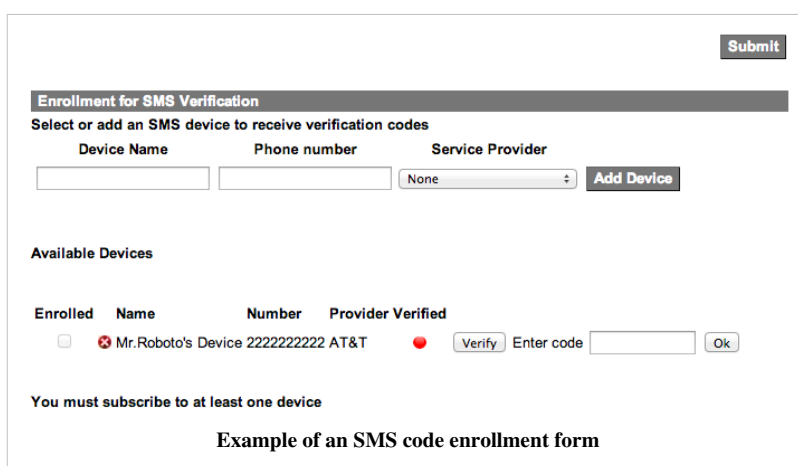
If the code is correct, the device is automatically enrolled.

To enroll using SMS:

1. Navigate to **Password Reset > Enrollment**.
2. Select the **Enrolled** check box for one or more SMS enabled devices.

If needed, fill out the security question form.

3. Click **Submit**.



The screenshot shows the 'Enrollment for SMS Verification' form. At the top right is a 'Submit' button. Below the title bar, the instruction 'Select or add an SMS device to receive verification codes' is displayed. There are three input fields: 'Device Name', 'Phone number', and 'Service Provider' (with a dropdown menu currently set to 'None'). An 'Add Device' button is to the right of the 'Service Provider' field. Below these fields is a section titled 'Available Devices'. It contains a table with columns: 'Enrolled', 'Name', 'Number', and 'Provider Verified'. One device is listed: 'Mr.Roboto's Device' with number '2222222222' and provider 'AT&T'. The 'Enrolled' checkbox is unchecked. To the right of the device name is a red delete icon. To the right of the number is a red 'Verify' button. To the right of the provider is an 'Enter code' input field and an 'Ok' button. Below the table, a message states 'You must subscribe to at least one device'. At the bottom, the text 'Example of an SMS code enrollment form' is centered.

After you have verified a device, it can be enrolled or unenrolled as needed by clearing the **Enrolled** check box. Clicking the delete icon removes the device from the enrollment form and from your user profile. You can add and enroll multiple devices.

## References

- [1] [https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c\\_EnrollInPasswordReset.html](https://docs.servicenow.com/bundle/jakarta-it-service-management/page/administer/login/concept/c_EnrollInPasswordReset.html)

# Resetting Passwords

---



**Note:** This article applies to Fuji and earlier releases. For more current information, see *Password Reset and Password Change Applications* <sup>[1]</sup> at <http://docs.servicenow.com> **The ServiceNow Wiki is no longer being updated. Visit <http://docs.servicenow.com> for the latest product documentation.**

## Overview

The specific details of the password reset process vary depending on the verification type and settings you have chosen. Also, depending on your organization's password reset requirements, users can reset their passwords through a publicly accessible web page, or service desk employees with the `password_reset_service_desk` role can perform password resets on behalf of users.

To see a video on resetting a user's password, go to [Resetting User Passwords \(Video\)](#) <sup>[4]</sup>.

## Unblocking a user whose password was reset by Service Desk

A user can be blocked (not allowed to use the Password Reset application) if the user is flagged as exceeding max reset attempts or is manually banned. When the password reset request is made through self-service, the user is unblocked when the password is successfully reset. In Dublin, a blocked user is called a "locked out" user.

In contrast, Service Desk users are allowed any number of re-tries -- `max_attempt_reached` does not apply and the "blocked" status for the requester is not reset after the password is reset. As a result, after you (the Service Desk admin) have successfully helped to reset a requester's password, you must manually unblock the requester. If you do not unblock the user, then the system resets the blocked state only when the `max_attempt_window` time period has elapsed and the user tries to reset the password using self-service. To unblock a user:

1. Navigate to **Password Reset > Blocked Users (Password Reset > Locked-out Users in Dublin)** to access the `pwd_user_lockout` table.
2. Change the **Blocked** setting (**Locked** in Dublin) for the user from true to false.

## Example Password Reset Scenario

The following example shows how service desk employees can reset passwords using email verification with unlock user account enabled.

1. Go to **Password Reset > Service Desk**.
2. Select a user and the password reset process.

The screenshot shows a web form titled "Password Reset Assistance (Demo Self-Service Process 1)" with the instruction "Enter your identification information". At the top, there is a progress bar with three steps: "Create Request" (highlighted in green), "Verify Identity", and "Reset password". The form contains two input fields: "Email address:" with the value "andrew.jackson@example.com" and "Type the characters you see in the image below:" with the value "nfh3r". Below the second field is a CAPTCHA image showing the characters "nfh3r" with a cursor pointing to the 'h'. To the right of the CAPTCHA is a "Replace image" button. At the bottom right is a "Verify Identity" button. The text "Password reset create request" is centered at the bottom of the form.

3. Click **Verify Identity**.
4. In the Verify Identity form, accept or reject the answer provided by the user and click **Continue**.

The screenshot shows a web form titled "Security Questions Identity Verification" with the instruction "Answer the following questions". At the top, there is a progress bar with three steps: "Create Request", "Verify Identity" (highlighted in green), and "Reset password". The form contains three questions, each with a text input field: "What is the name of the first school you attended?" (value: "\*\*\*\*\*"), "What was the make and model of your first car?" (value: "\*\*\*\*\*"), and "What is the country of your ultimate dream vacation?" (value: "\*\*\*\*\*"). At the bottom right is a "Continue" button. The text "Password reset verify identity" is centered at the bottom of the form.

Starting with the Eureka release, if you have accepted the user's identity, the Reset Password form appears with the status for the user—whether they were successfully verified and their current account lock state.

- If the user is not locked, the form displays the **Reset password** button. To change the user's password, enter the new password, and click

### Reset Password.

- If the user is locked, the form displays the **Reset password** and **Unlock account** buttons. You can enter a new password and click **Reset password** to reset the password and automatically unlock the account. If **Enable account unlock** is selected for the password process, you can unlock the account without resetting the password by clicking **Unlock account**.


Create Request

Verify Identity

Reset password

Create Password

Identity successfully verified

Current account lock state: 

**Create a new password.**

- At least 8 characters
- At least one uppercase and one lower case letter
- At least one number

New Password:

Retype Password:

[Reset password](#)

Password reset change password

## References

- [1] <https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/login/reference/password-reset-landing-page.html>

---

# Administration

---

## Monitoring

---

### Overview

The Password Reset application provides a number of tools for monitoring and troubleshooting password reset activities. Users with the `password_reset_credential_manager` or `password_reset_admin` role can view and quickly determine the status of password reset activities, identify potential security threats, and monitor for compliance with the organization's password security policies.

### Using the Overview Module

The **Password Reset > Overview** module displays the following password reset reports:

- **Password Requests (last 7 days):** number of password reset requests by type over the last 7 days.
- **Blocked Users (last 7 days):** number of users blocked by reason over the last 7 days. This is referred to as **Locked Users** in the Dublin release.
- **Password Reset Request Status (last 7 days):** current status of all password reset requests by process.
- **Password Reset Request by Action (last 30 days):** number of password reset requests by action type—Reset Password, Unlock Account, Reset and Unlock, or Change Password (Fuji release).
- **Password Reset Top Users (last 30 days):** number of password reset requests per user. A large number of password reset requests from a single user may indicate a security issue.
- **Password Reset Failed Verifications (last 7 days):** number of failed verification attempts, by verification instance. A failed verification occurs when a user attempts to reset their password, but fails for one reason or another, during the identity verification step. A large number of failed verification attempts for a specific type of verification may indicate that the process is too complicated or unclear.
- **Password Reset Enrollment By Verification:** number of users, by verification type, that have and have not enrolled in the password reset program. A large number of users who have not enrolled may indicate a compliance or communication issue within the organization.
- **Password Change Top Users (last 30 days):** number of password change requests per user. A large number of password change requests from a single user may indicate a security issue (Fuji release).

At certain intervals, ServiceNow automatically purges data that is used to generate reports. As a result, report information may change significantly immediately after a data purge. Contact ServiceNow Technical Support if you must modify purge intervals. For more information, see [Managing Data for Reports](#).

Users with the `password_reset_admin` role can customize the layout of the reports that appear in the Overview module.

---

## Viewing Password Reset Activities

The Reset Requests, Activity Log, and Blocked Users (Locked Users in the Dublin release) modules are useful for monitoring password reset activities and troubleshooting password reset issues. They also provide access to more detailed information than the Overview module offers.

### Reset Requests

The Reset Requests module displays the type and current status of each password reset and password change request (available starting with the Fuji release). Information about reset requests is stored in the Password Reset Request [pwd\_reset\_request] table.

To view reset requests, navigate to **Password Reset > Reset Requests**.

A password reset request consists of the following fields.

Field	Description
User	The user whose password is being reset or changed.
Process	The process managing this password reset request. See Configuring Processes.
Type	The type of password reset request: <ul style="list-style-type: none"> <li>• <b>Change Password:</b> This request is for a password change (Fuji release).</li> <li>• <b>Help Desk:</b> This reset request was opened on behalf of the requesting user by a service desk agent.</li> <li>• <b>Self Service:</b> This reset request was opened by the requesting user.</li> </ul>
Action	The corrective action to be taken by password reset request.
Type	<ul style="list-style-type: none"> <li>• <b>Change Password:</b> update the credential store with the new password (Fuji release).</li> <li>• <b>Reset and Unlock Account:</b> generate a new password for the user and unlock the user's account.</li> <li>• <b>Reset Password:</b> generate a new password for the user.</li> <li>• <b>Unlock Account:</b> unlock the user's account.</li> </ul>
Status	The result of the password reset request: <ul style="list-style-type: none"> <li>• <b>Completed With Failure:</b> user completed all of the steps in the password reset process, but the password was not reset in the credential store.</li> <li>• <b>Completed With Success:</b> user completed all of the steps in the password reset process and the password was reset in the credential store.</li> <li>• <b>Expired:</b> user did not complete all of the steps in the password reset process in the time allowed.</li> <li>• <b>In Progress:</b> user is currently working through the steps to reset the password.</li> <li>• <b>Max Number of Attempts:</b> user failed to correctly answer the security questions during the identity verification step and has exceeded the maximum number of attempts allowed.</li> <li>• <b>Verified:</b> user has completed the identity verification step and is verified. The user can now move to the password reset step.</li> </ul>
Active	Whether the request is open or closed.
Retry	The total number of times the user has attempted to complete a password reset request.



## Activity Log

The activity log provides detailed information that can be used for troubleshooting and for generating reports on password reset metrics. Information contained in the activity log is stored in the Password Reset Activity Log [pwd\_reset\_activity] table.

To view the activity log, navigate to **Password Reset > Activity Log**.

## Blocked Users

A blocked user, called a locked user in the Dublin release, is a person who is prevented from logging in to the ServiceNow system if any of the following events occur:

- The user exceeds the limit for the number of failed password attempts.
- The user's most recent password reset occurred before the wait time required until the next reset.
- The user fails to provide the correct information while attempting to reset their password.

If the number of blocked, or locked, users exceeds the limit within a defined time interval, it triggers a system log event. You can configure the number of blocked, or locked, users and the time interval required to generate the log event by setting the `password_reset.activity_monitor.incident_threshold` and `password_reset.activity_monitor.incident_window` properties.

To view a list of blocked users, navigate to **Password Reset > Blocked Users**. For the Dublin release, to view a list of locked users, navigate to **Password Reset > Locked Users**.

## Unblocking Users

Only the password reset administrator or a service desk employee can unblock a user.

1. Navigate to **Password Reset > Blocked Users**.
2. Select a user from the list of blocked users.
3. Select **Delete** from the **Actions on selected rows** list.

For the Dublin release, to unlock a user, navigate to **Password Reset > Locked Users**, select a user from the list of locked users, and select **Delete** from the **Actions on selected rows** list.

## Subscribing to Blocked User Notifications

As a password reset administrator, you can receive email notifications when the number of users who are blocked or locked exceeds the password blocked threshold (default 10). Notifications can alert you to suspicious activities.

1. Add a new email notification device or modify an existing device.
2. Subscribe to the **Password Reset-Activity Monitor Lockout** notification.

## Managing Data for Reports

At specific intervals, ServiceNow purges data that is used for password reset monitoring and reporting. Information contained in reports and monitoring tools may change dramatically immediately after a data purge. Contact ServiceNow Technical Support if you need to modify purge intervals.

Table Name	Purge Interval
pwd_reset_request	90 days (7,776,000 seconds). Depending on your organization's data monitoring requirements, you can request that this rule be configured to purge successful requests after 90 days and keep failed requests for 120 days.
pwd_user_lockout	90 days (7,776,000 seconds). Depending on your organization's data monitoring requirements, you can request that this rule be configured to purge successful requests after 90 days and keep failed requests for 120 days.
pwd_reset_activity	90 days (7,776,000 seconds).
pwd_activity_monitor	90 days (7,776,000 seconds).
pwd_dvc_enrollment_code	1 day (86,400 seconds).
pwd_sms_code	1 day (86,400 seconds).

# Properties

## Overview

The Properties module provides access to all configurable properties for the Password Reset application. Users with the password\_reset\_admin role can set these properties.

## Setting Properties

While there are no range limits for the values you can enter for properties, consider using only positive integer values starting at 1. When determining the limit for the upper range of a property, consider the task that the user is performing. For example, you would not want to provide users who are attempting to verify their identity with 100 attempts. A more common value is 3 attempts. Similarly, you might not want to force users who are completing the enrollment process to spend time selecting and answering 30 security questions. The more commonly used number of security questions is between 5 and 7.

To set password reset properties:

1. Navigate to **Password Reset > Properties**.
2. Fill in the fields, as appropriate (see table).
3. Click **Save**.

Name	Description
<b>Password Reset Global Properties</b>	
password_reset.wf.timeout	Determines the maximum wait time, in milliseconds, for the workflow to execute. The workflow is triggered during the password reset request when the user clicks <b>Submit</b> . <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 500 (milliseconds)</li></ul>
password_reset.wf.refresh_rate	Determines how often to check status of the workflow. <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 90000 (milliseconds)</li></ul>
password_reset.captcha.ignore	Enables or disables captcha functionality. <ul style="list-style-type: none"><li>• <b>Type:</b> true/false</li><li>• <b>Default value:</b> false</li></ul>

### Password Reset Request Properties

password_reset.request.max_attempt	<p>Determines the number of password reset attempts a user has before they are locked out for a period determined by the value in <code>max_attempt_window</code>.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 3 (attempts)</li> </ul>
password_reset.request.max_attempt_window	<p>Determines how long a user is blocked or prevented from changing their password after trying the maximum number of times.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 1440 (minutes)</li> </ul>
password_reset.request.expiry	<p>Determines how long before a password reset request expires.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 10 (minutes)</li> </ul>
password_reset.request.success_window	<p>Determines how long a user must wait to reset their password again after they have successfully reset their password.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 1440 (minutes)</li> </ul>
password_reset.request.unlock_window	<p>Determines how long a user must wait after a successful unlock operation before starting a new request. This property is available starting with the Eureka release.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 1440 (minutes)</li> </ul>

### Password Reset Security Question Properties

password_reset.qa.num_reset	<p>Specifies the number of questions a user has to answer in order to verify their identity during the password reset process.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 3 (questions)</li> <li>• <b>Possible values:</b> integers that are less than the number specified for the <code>num_enroll</code> property.</li> <li>• <b>Learn more:</b> <a href="#">Configuring Verifications</a></li> </ul> <p><b>Note:</b> You can override this security question property by adding the <code>num_reset</code> parameter in the security question verification.</p>
password_reset.qa.num_enroll	<p>Specifies the number of questions a user has to select and answer to be enrolled in the password reset program.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 5 (questions)</li> <li>• <b>Learn more:</b> <a href="#">Configuring Verifications</a></li> </ul> <p><b>Note:</b> You can override this security question property by adding the <code>num_enroll</code> parameter in the security question verification.</p>

### Password Reset SMS Code Properties

password_reset.sms.max_per_day	<p>Determines the maximum number of SMS codes that are sent to a user within one 24 hour period. When a user clicks the <b>Send Verification Code</b> button, the 24 hour period begins.</p> <ul style="list-style-type: none"> <li>• <b>Type:</b> integer</li> <li>• <b>Default value:</b> 10 (per day)</li> <li>• <b>Learn more:</b> <a href="#">Configuring Verifications</a></li> </ul> <p><b>Note:</b> You can override this SMS code property by adding the <code>max_per_day</code> parameter in the SMS code verification.</p>
--------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

password_reset.sms.pause_window	<p>Determines the amount of time that needs to pass before another SMS code can be sent to a user.</p> <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 2 (minutes)</li><li>• <b>Learn more:</b> Configuring Verifications</li></ul> <p><b>Note:</b> You can override this SMS code property by adding the <code>pause_window</code> parameter in the SMS code verification.</p>
password_reset.sms.default_complexity	<p>Specifies the number of characters required for a user to reset their password.</p> <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 4 (digits)</li><li>• <b>Learn more:</b> Configuring Verifications</li></ul> <p><b>Note:</b> You can override this SMS code property by adding the <code>complexity</code> parameter in the SMS code verification.</p>
password_reset.sms.expiry	<p>Determines the amount of time, in minutes, until the SMS code sent to the user expires.</p> <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 5 (minutes)</li><li>• <b>Learn more:</b> Configuring Verifications</li></ul> <p><b>Note:</b> You can override this SMS code property by the <code>expiry</code> parameter in the SMS code verification.</p>

#### Password Reset Monitoring and Reporting Properties

password_reset.activity_monitor.incident_window	<p>Determines the time window to count the number of blocked users (locked users in the Dublin release).</p> <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 60 (minutes)</li></ul>
password_reset.activity_monitor.incident_threshold	<p>Specifies the number of blocked (or locked) users, within the given time window, that triggers a system log event.</p> <ul style="list-style-type: none"><li>• <b>Type:</b> integer</li><li>• <b>Default value:</b> 10 (blocked users)</li></ul>

# CMS Integration

## Overview

Administrators can configure ServiceNow content management (CMS) to define a single-site access point that includes the password reset service. For example, you may want to create an employee self-service site that provides password reset service. Each password reset process requires a separate CMS page.

## Integrating Password Reset

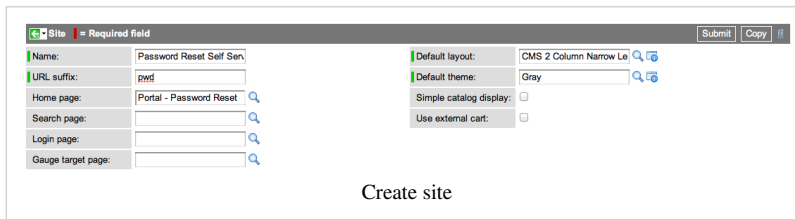
To integrate password reset with content management:

1. Go to **Content Management > Specialty Content > iFrames** and create a new iFrame record.



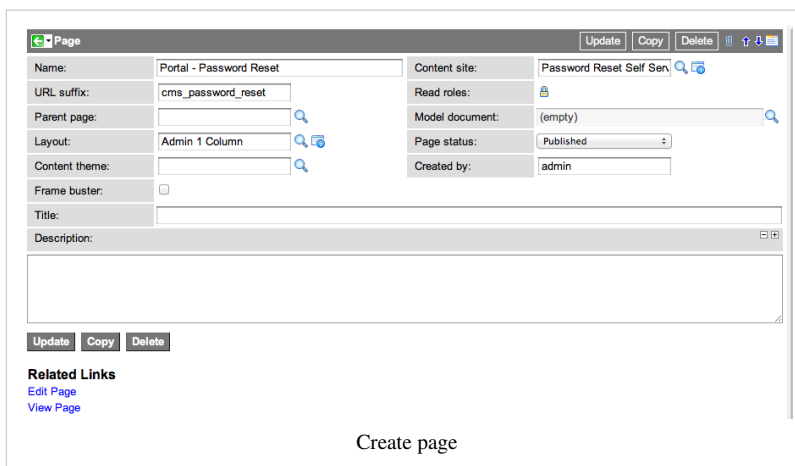
Create iframe

2. Go to **Content Management > Sites** and create a site that has no login page.



Create site

3. Go to **Content Management > Pages** and create a password reset page.



Create page

4. Enter the site created above—for this example, Password Reset Self-Service—in the **Content site** field.
5. Right-click the form header and click **Save**.
6. Click **Edit Page** under **Related Links** and then click **Add content**.
7. Under the **Content Blocks** section, select the iframe added in the previous steps. Add any additional content.

# Installed Components

---

## Overview

The following components are installed with the Password Reset plugin:

- Tables
- Workflows
- Script Includes
- Business Rules
- User Roles
- Web Services

The Password Reset Orchestration Add-on provides additional options within the Password Reset application. However, no additional components are installed.

## Tables

Table Name	Description
Password Reset Active Answer [pwd_active_answer]	Security questions and associated answers, in an encrypted state, that users have selected while going through the enrollment process.
Password Reset Active Question [pwd_active_question]	Security questions that users have selected while going through the enrollment process.
Password Reset Activity Log [pwd_reset_activity]	All password reset requests.
Password Reset Activity Monitor [pwd_activity_monitor]	Password reset lockout activity.
Password Reset Credential Store [pwd_cred_store]	Password reset credential stores that are available.
Password Reset Credential Store Parameters [pwd_cred_store_param]	User-created credential store parameters.
Password Reset Credential Store Types [pwd_cred_store_type]	Password reset credential store types that are available.
Password Reset Device Enrollment Code [pwd_dvc_enrollment_code]	Device enrollment codes that have been sent to users during SMS code enrollment.
Password Reset Devices [pwd_device]	User SMS devices that are in a state of verified.
Password Reset Enrollment for Verification [pwd_enrollment]	Information about user enrollment by verification.
Password Reset Enrollment Snapshot [pwd_enrollment_snapshot]	Snapshot of user enrollment by verification. This table is regenerated daily by a scheduled job named Password Reset Enrollment Snapshot.
Password Reset Extension Type [pwd_extension_type]	Extension types that are available.
Password Reset Identification Type [pwd_identification_type]	Password reset identification types that are available.
Password Reset Process [pwd_process]	Password reset processes that are available.
Password Reset Process Credential Store [pwd_map_proc_to_cred_store]	Credential stores and the associated password reset processes that the application is using.
Password Reset Process User Group [pwd_map_proc_to_group]	Groups and the associated password reset processes that the application is using.

Password Reset Process Verification [pwd_map_proc_to_verification]	Verifications and the associated password reset processes that the application is using.
Password Reset Question [pwd_question]	Questions that the application uses for security question verifications.
Password Reset Request [pwd_reset_request]	Information about password reset requests.
Password Reset SMS Verification Code [pwd_sms_code]	SMS verification codes that have been sent to users for a password reset.
Password Reset User Lockout [pwd_user_lockout]	Users that are locked out of password reset.
Password Reset Verification [pwd_verification]	Verifications that are available.
Password Reset Verification Param [pwd_verification_param]	User-created verification parameters.
Password Reset Verification Type [pwd_verification_type]	Password reset verification types that are available.

## Workflows

### Password Reset

Workflow Name	Description
Pwd Reset - Master	Master workflow that is used to reset a password.
Pwd Reset - AD	Subflow called by the master password reset workflow that resets passwords on Microsoft Active Directory credential stores.
Pwd Reset - Local ServiceNow	Subflow called by the master password reset workflow that resets passwords on local ServiceNow instance credential stores.
Pwd Reset - Remote ServiceNow	Subflow called by the master password reset workflow that resets passwords on remote ServiceNow instance credential stores.
Pwd Reset - Mock Success	Mock subflow called by the master password reset workflow that emulates a successful password reset process.
Pwd Reset - Mock Fatal Mock Workflow	Mock subflow called by the master password reset workflow that emulates a fatal password reset process.
Pwd Reset - Mock Non Fatal	Mock subflow called by the master password reset workflow that emulates a non-fatal password reset process.

### Password Change

The password change workflows are available starting with the Fuji release.

Workflow Name	Description
Pwd Change - Master	Master workflow used to change a password.
Pwd Change - AD	Subflow called by the master password change workflow that changes passwords on Microsoft Active Directory credential stores.
Pwd Change - Local ServiceNow	Subflow called by the master password change workflow that changes passwords on local ServiceNow instance credential stores.
Pwd Change - Remote ServiceNow	Subflow called by the master password change workflow that changes passwords on remote ServiceNow instance credential stores.

## Connection Test

Workflow Name	Description
Pwd Connection Test - Master	Master workflow that tests credential store connectivity.
Pwd Connection Test - AD	Subflow called by the master connection test workflow that tests a connection to an Active Directory credential store.
Pwd Connection Test - Local SN	Subflow called by the master connection test workflow that tests a connection to a local ServiceNow instance credential store.
Pwd Connection Test - Remote SN	Subflow called by the master connection test workflow that tests a connection to a remote ServiceNow instance credential store.
Pwd Connection Test - Mock Failure	Mock workflow called by the master connection test workflow that imitates a failure.
Pwd Connection Test - Mock Success	Mock workflow called by the master connection test workflow that imitates a success.

## Unlock

Workflow Name	Description
Pwd Get Lock State - Master	Master workflow that determines if a user account is locked out.
Pwd Get Lock State - AD	Subflow called by the master get lock state workflow that queries user accounts in Activity Directory credential stores.
Pwd Get Lock State - Local SN	Subflow called by the master get lock state workflow that queries user accounts in local ServiceNow instance credential stores.
Pwd Get Lock State - Remote SN	Subflow called by the master get lock state workflow that queries user accounts in remote ServiceNow instance credential stores.
Pwd Unlock Account - Master	Master workflow that unlocks locked user accounts.
Pwd Unlock Account - AD	Subflow called by the master unlock account workflow that unlocks user accounts in Activity Directory credential stores.
Pwd Unlock Account - Local SN	Subflow called by the master unlock account workflow that unlocks user accounts in local ServiceNow instance credential stores.
Pwd Unlock Account - Remote SN	Subflow called by the master unlock account workflow that unlocks user accounts in remote ServiceNow instance credential stores.



## Script Includes

Script Name	Description
PwdAjaxChangePassword	Processes AJAX requests to change passwords.
PwdAjaxSMSProcessor	Processes AJAX SMS code generation requests.
PwdVerifySimpleProcessor	Processes the verification form request, and returns a value indicating if the user was verified.
PwdVerifyPersonalDataProcessor	Compares the answers provided by the user with the data in the system.
PwdAjaxVerifyIdentityServiceDesk	Supports the create request UI for the service desk module in password reset.
PwdAjaxEnrollmentProcessor	Enrolls the user for the specified verification.
PwdDefaultAutoGenPassword	Generates a password based on a random word and four digits.
PwdTestCredStoreConnectionWorker	Provides the credential store connection test.
PwdVerifyQuestionsProcessor	Processes user input for security question verifications.
PwdAjaxPublicEnrollSMS	Sends the details of the subscription or mobile from the user profile.
PwdAjaxRequestProcessor	Processes AJAX requests for the Password Reset application.
PwdAjaxSMSProcessor	Processes AJAX SMS code generation requests.
PwdQuestionsEnrollmentCheck	Checks whether the user is enrolled with the specified verification.
PwdAlwaysEnrolled	Checks if the user is enrolled. This is a default script that always returns true for <code>isEnrolled()</code> .
PwdVerificationParameterUtility	Provides methods for various verification parameter related business rules.
PwdEnrollSMSProcessor	Processes enrollment for SMS code verifications.
PwdEnrollQuestionsProcessor	Processes enrollment for question and answer for verifications.
PwdMockIsEnrolled	Checks whether the user is enrolled in a specific verification.
PwdAjaxEnrollSMS	Manages mobile device subscriptions for SMS code verifications.
PwdEnrollSampleProcessor	Processes enrollment for sample verifications.
PwdVerifyPersonalDataConfirmationProcess	Verifies that the answer was accepted by the user.
PwdAjaxVerifyIdentity	Verifies the information provided by the user during the first stage of a password reset request.
PwdSMSEnrollmentCheck	Checks whether a user is enrolled in a specific verification. Returns a boolean value indicating if the user is enrolled.
PwdVerifyUser	Checks if user exists.
PwdDefaultUserAccountLookup	Utilizes a user's <code>sys_user</code> id to find their account in a credential store. This is the default script for user account lookup. The default mapping is to use <code>user_name</code> as the account name.
PwdNotificationHelper	Provides methods for common notification related tasks.
PwdAjaxWFRequestProcessor	Processes workflow related tasks for the Password Reset application.
PwdVerifySMSProcessor	Processes user input for SMS code verification.
PwdAjaxVerifyProcessor	Checks if the user is verified by all verification methods.
PwdIdentifyViaEmail	Looks up users by their email.
PwdPostProcessor	Executes actions after the process completes.
PwdIdentifyViaUsername	Looks up users by their user name. This is an identity extension.

## UI Scripts

Name	Description
pwd_csrf_common_ui_script	Handles the password reset cross-site request forgery token.
pwd_enrollment_submit_event	Responds to the submit event for password reset enrollment.
pwd_enroll_questions_ui	Generates the enrollment form for security question verifications.
pwd_enroll_sample_ui	Generates the enrollment form for mock security question verification.
pwd_enroll_sms_ui	Generates the enrollment form for SMS verifications.

## Business Rules

Business Rule	Table	Description
Verify Account Lookup Script	Password Reset Credential Store [pwd_cred_store]	Checks if the account lookup script has the correctly named function.
Prevent against deletion	Password Reset Credential Store [pwd_cred_store]	Checks if the credential store is part of an active process before allowing deletion.
Send SMS code	Password Reset Device Enrollment Code [pwd_dvc_enrollment_code]	Sends an enrollment code to a device.
Prevent against deletion	Password Reset Identification Type [pwd_identification_type]	Prevents an identification type from being deleted if it is part of an active process.
Single credential store per process	Password Reset Process Credential Store [pwd_map_proc_to_cred_store]	Prevents having more than one credential store per process.
Deactivate process with no group	Password Reset Process User Group [pwd_map_proc_to_group]	Deactivates the process if it does not apply to all users or if the groups associated with it are removed.
Check unique verifications	Password Reset Process Verification [pwd_map_proc_to_verification]	Prevents a verification from being assigned multiple times to a specific password reset process.
Deactivate process with no verification	Password Reset Process Verification [pwd_map_proc_to_verification]	Deactivates the process if the verifications associated with it are removed.
Password Reset Validate Auto-generate	Password Reset Process [pwd_process]	Checks that either <b>Email password</b> or <b>Display password</b> is selected when the <b>Auto-generate password</b> check box is selected.
Validate Process	Password Reset Process [pwd_process]	Verifies that a password reset process is configured correctly.
Update proc_to_cred_store	Password Reset Process [pwd_process]	Enforces a one-to-one relation between a password reset process and a credential store.
Set new record flag	Password Reset Process [pwd_process]	Sets a new record flag for the client to take appropriate action.
Validate Security Question	Password Reset Question [pwd_question]	Validates rules for security questions such as no duplicates or empty questions.
Password Reset Activity Monitor	Password Reset User Lockout [pwd_user_lockout]	Creates an event when the number of users locked out of password reset during a specific interval exceeds the threshold value.
Add default parameters QA verification	Password Reset Verification [pwd_verification]	Generates parameters for security question verifications if none are specified.
Add params personal confirm verification	Password Reset Verification [pwd_verification]	Generates parameters for personal data confirmation verifications if none are specified.

Add params personal verification	Password Reset Verification [pwd_verification]	Generates parameters for personal data verification if none are specified.
Prevent against deletion	Password Reset Verification [pwd_verification]	Prevents a verification from being deleted if it is part of an active process.
Add default parameters SMS verification	Password Reset Verification [pwd_verification]	Generates parameters for SMS code verifications if none are specified.
Parameter Names Cannot Be Updated	Password Reset Verification Param [pwd_verification_param]	Prevents parameter name changes.
Personal Data Param Validation	Password Reset Verification Param [pwd_verification_param]	Checks that a column exists in the sys_user table for the parameter used in a personal data verification.
Security Questions Param Validation	Password Reset Verification Param [pwd_verification_param]	Checks for valid parameters in security question verifications.
Personal Data Confirm Param Validation	Password Reset Verification Param [pwd_verification_param]	Checks that a column exists in the sys_user table for the parameter used in a personal data confirmation verification.
SMS Code Param Validation	Password Reset Verification Param [pwd_verification_param]	Checks for valid parameters in SMS code verifications.
VerifyAutoEnroll	Password Reset Verification Type [pwd_verification_type]	Checks if auto-enroll is selected and ensures that an enrollment check script is provided.

## User Roles

Role	Contains Roles	Description
password_reset_admin	password_reset_credential_manager script_include_admin ui_macro_admin workflow_admin	The password reset administrator creates password reset processes, verification methods, and credential stores.
password_reset_credential_manager	password_reset_service_desk script_include_admin workflow_admin	The credential store manager monitors password reset activity through reports in the Overview module. Creates credential store stores.
password_reset_service_desk		The service desk employee responds to password reset requests by phone or email. Verifies identity of person requesting a password reset. Generates temporary passwords and assists users with resetting their passwords.

## Web Services

## SOAP Messages

SOAP Message	Description
Change Password	When the Orchestration Add-on plugin is active, the system can use the SOAP protocol to change passwords on remote credential stores such as a remote ServiceNow instance.
Password Reset Request	When the Orchestration Add-on plugin is active, the system can use the SOAP protocol to reset passwords on remote credential stores such as a remote ServiceNow instance.

## Scripted Web Services

Scripted Web Service	Description
ChangePwd	When the Orchestration Add-on plugin is active, the system can use this scripted Web Service to change passwords on remote credential stores such as a remote ServiceNow instance.

---

# Password Reset Extensions

---

## Scripting

---

### Overview

Password reset scripting enables you to customize password reset by creating your own credential store, verification, and identification types, and extend them by defining extension scripts. The easiest way to customize password reset is to create your custom types and scripts, and then follow the configuration steps described in Password Reset, selecting the new types you have created. The following components can be customized.

- **Credential store types:** Define new types for how to connect to your credential stores by creating custom workflows for connection and testing.
- **Verification types:** Define new types for how users are verified.
- **Identification types:** Define new types for how users can identify themselves.

Password reset scripting is available to users with the password\_reset\_admin role.

### Extension Scripts

Extension scripts allow you to extend password reset functionality by defining custom scripts that can be used in a credential store, verification, or identification type, or as a post-processor in a process. Extension scripts are predefined hooks within the Password Reset application that perform specific types of behavior defined by the extension category, which refers to where the script will be used. For details on how to define extension scripts, see Creating Extension Scripts.

### Example

The following example shows a script that performs a user account lookup and processes an Identification form. The main script calls two extension scripts, one to perform the user account lookup, and the other to process the Identification form.

```
// User account lookup
var lookupExtensionSysId =
getExtensionScriptSysId('SampleUserAccountLookupExtension',
'user_account_lookup');
var lookupExtension = new SNC.PwdExtensionPoint(lookupExtensionSysId);

// Setup parameters required for this extension type - userId
var params = new SNC.PwdExtensionPointParameter();
params.userId= 'joe.employee';

// Invoke the extension
var answer = lookupExtension.process(params);
gs.print('user: ' + answer);
```

```

//Form processor sample - Identification form processor
var identExtensionSysId =
getExtensionScriptSysId('SampleIdentificationProcessorExtension',
'identification_form_processor');
var identificationExtension = new
SNC.PwdExtensionPoint(identExtensionSysId);

// Setup parameters required for this extension type - processId
var params = new SNC.PwdExtensionPointParameter() ;
params.processId = 'pwdreq1234';

// Simulate the posted form parameter for the identification processor
var request = new SNC.PwdExtensionPointParameter() ; // A real life
case will inject it's own request object
request.setParameter('sysparm_user_id', 'joe.employee');

var userIdentity = identificationExtension.processForm(params,
request);
gs.print('identity: ' + userIdentity);

// Simple helper to return the sys-id for a given extension script
function getExtensionScriptSysId(scriptName, category) {
    var result;
    var gr = new GlideRecord('sys_script_include');
    gr.addQuery('name', scriptName);
    gr.addQuery('script', 'CONTAINS', 'category:
\'password_reset.extension.' + category + '\');
    gr.query();

    if (gr.next() ) {
        result = gr.getValue('sys_id');
    }
    return result;
}

```

The following is an example of an extended process function in the User Account Lookup category used to define a credential store. To create this extension script, go to **Password Reset > Extensions > New extension script** and create a new script as described in [Creating Extension Scripts](#). To configure the User Lookup in a password reset process, see [Configuring Credential Stores](#).

```

<?xml version="1.0" encoding="UTF-8"?>
<record_update table="sys_script_include">
  <sys_script_include action="INSERT_OR_UPDATE">
    <active>true</active>
    <client_callable>false</client_callable>
    <description>Simple account lookup that returns the supplied user id</description>
    <name>SampleUserAccountLookupExtension</name>
    <script><![CDATA[var SampleUserAccountLookupExtension = Class.create();

```

```

SampleUserAccountLookupExtension.prototype = {
  category: 'password_reset.extension.user_account_lookup',      // DO
NOT REMOVE THIS LINE!

  /*****
   * Returns the credential-store account id for a given user
   *
   * @param params.userId The sys-id of the user being checked (table:
sys_user)
   * @return              The credential-store account-id (string) for
a given user
   *****/
  process: function(params) {
    return params.userId;
  },

  type: 'SampleUserAccountLookupExtension'
};]]</script>

<sys_created_by>admin</sys_created_by>
<sys_created_on>2013-07-30 16:44:55</sys_created_on>
<sys_id>2df5a103d73201002bb9af728e610333</sys_id>
<sys_mod_count>1</sys_mod_count>
<sys_updated_by>admin</sys_updated_by>
<sys_updated_on>2013-07-30 16:46:00</sys_updated_on>
</sys_script_include>
<sys_app_file action="INSERT_OR_UPDATE">
  <customer_update>false</customer_update>
  <publish_override/>
  <replace_on_upgrade>false</replace_on_upgrade>
  <restore/>
  <sys_app/>
  <sys_code>!!1W4</sys_code>
  <sys_created_by>admin</sys_created_by>
  <sys_created_on>2013-07-30 16:44:55</sys_created_on>
  <sys_id>8306e143d73201002bb9af728e6103d3</sys_id>
  <sys_mod_count>0</sys_mod_count>
  <sys_name>SampleUserAccountLookupExtension</sys_name>
  <sys_parent/>
  <sys_path>!!1W4</sys_path>
  <sys_policy/>
  <sys_source_deleted>false</sys_source_deleted>
  <sys_source_id>2df5a103d73201002bb9af728e610333</sys_source_id>
  <sys_source_table>sys_script_include</sys_source_table>
  <sys_type>code</sys_type>
  <sys_update_name>sys_script_include_2df5a103d73201002bb9af728e610333</sys_update_name>
  <sys_updated_by>admin</sys_updated_by>

```

```
<sys_updated_on>2013-07-30 16:46:00</sys_updated_on>
</sys_app_file>
</record_update>
```

The following is an example of an extended processForm function in the Identification Form Processor category that can be used to create an identification type. To create this extension script, go to **Password Reset >Extensions > New extension script** and create a new script as described in Creating Extension Scripts. To define the identification type, go to Defining Identification Types. To configure the identification type in a password reset process, see Configuring Processes.

```
<?xml version="1.0" encoding="UTF-8"?>
<record_update table="sys_script_include">
  <sys_script_include action="INSERT_OR_UPDATE">
    <active>true</active>
    <client_callable>false</client_callable>
    <description>Script that processes an identification form.&#13;
Returns the sys-id of the user that corresponds to the requested input;
if no user was found, null should be returned.&#13;
</description>
    <name>SampleIdentificationProcessorExtension</name>
    <script><![CDATA[var SampleIdentificationProcessorExtension = Class.create();
SampleIdentificationProcessorExtension.prototype = {
  category: 'password_reset.extension.identification_form_processor',
    // DO NOT REMOVE THIS LINE!

  /*****
   * Process the identification form request, and returns the user's
sys_id. if user was not identified return null.
   *
   * @param params.processId The sys-id of the calling password-reset
process (table: pwd_process)
   * @param request The form request object. fields in the
form can be accessed using: request.getParameter('<element-id>')
   *
Supported request paramters:
   *
sysparm_user_id - the user
identifier value entered in the form.
   * @return The sys-id of the user that corresponds to the requested
input; if no user was found, null should be returned.
   *****/
  processForm: function(params, request) {
    return request.getParameter('sysparm_user_id') + '_' +
params.processId;
  },

  type: 'SampleIdentificationProcessorExtension'
};]]></script>
    <sys_created_by>admin</sys_created_by>
```



```
<sys_created_on>2013-07-30 17:00:28</sys_created_on>
<sys_id>3a79a503d73201002bb9af728e610349</sys_id>
<sys_mod_count>1</sys_mod_count>
<sys_updated_by>admin</sys_updated_by>
<sys_updated_on>2013-07-30 17:08:41</sys_updated_on>
</sys_script_include>
<sys_app_file action="INSERT_OR_UPDATE">
  <customer_update>false</customer_update>
  <publish_override/>
  <replace_on_upgrade>false</replace_on_upgrade>
  <restore/>
  <sys_app/>
  <sys_code>!!1W5</sys_code>
  <sys_created_by>admin</sys_created_by>
  <sys_created_on>2013-07-30 17:00:28</sys_created_on>
  <sys_id>4799ed03d73201002bb9af728e610333</sys_id>
  <sys_mod_count>0</sys_mod_count>
  <sys_name>SampleIdentificationProcessorExtension</sys_name>
  <sys_parent/>
  <sys_path>!!1W5</sys_path>
  <sys_policy/>
  <sys_source_deleted>false</sys_source_deleted>
  <sys_source_id>3a79a503d73201002bb9af728e610349</sys_source_id>
  <sys_source_table>sys_script_include</sys_source_table>
  <sys_type>code</sys_type>
  <sys_update_name>sys_script_include_3a79a503d73201002bb9af728e610349</sys_update_name>
  <sys_updated_by>admin</sys_updated_by>
  <sys_updated_on>2013-07-30 17:08:41</sys_updated_on>
</sys_app_file>
</record_update>
```

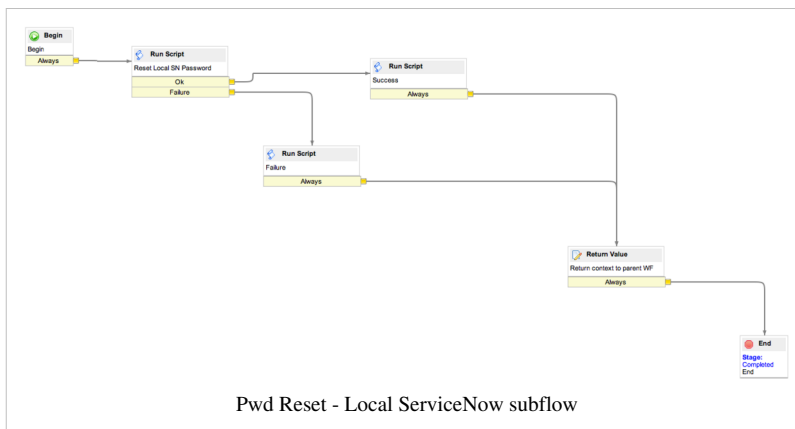
# Defining Credential Store Types

## Overview

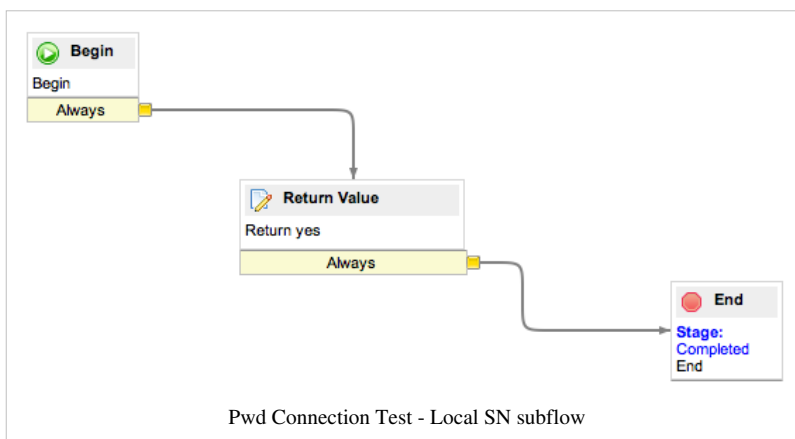
A credential store type is a set of workflows that specify how to connect to credential stores that contain information such as user names and passwords. For information on how to use credential store types, see [Configuring Credential Stores](#).

## Anatomy of a Credential Store Type

A credential store type requires a subflow that defines how to connect to the store, and can include an optional subflow that defines how to test the connection. The Pwd reset – AD and Pwd Reset - Local ServiceNow subflows are available as models for defining your own connection workflows. The following image shows the Pwd Reset - Local ServiceNow subflow.



The Pwd Connection Test - Local SN subflow, shown below, is available as a model for defining your own connection test workflows.



## Installed Credential Store Type

ServiceNow installs these example credential store types that you can use as models when creating your own credential store types.

Name	Description
Local ServiceNow Instance	Represents the current (local) ServiceNow instance.
AD Credential Store	Represents Active Directory credential store. (Installed with the Orchestration Add-on.)
Remote (SOAP) ServiceNow	Represents a remote ServiceNow instance. (Installed with the Orchestration Add-on.)

## Installed Workflows

ServiceNow installs these password reset workflows that you can use as an examples to create your own. For information on how to define workflows, see [Creating a Workflow](#).

### Password Reset Workflows

The following password reset workflows are used for connecting to a credential store.

Workflow	Description
Pwd Reset - Local ServiceNow	The current (local) ServiceNow instance.
Pwd Reset - Master	The password reset master workflow.
Pwd Reset - Mock Fatal	An example workflow to use in password reset testing to simulate a fatal error. No retries.
Pwd Reset - Mock Non Fatal	An example workflow to use in password reset testing to simulate a non-fatal error.
Pwd Reset - Mock Success	An example workflow to use in password reset testing to simulate a successful completion.

### Connection Test Workflows

The following workflows are used for performing a connection test.

Workflow	Description
Pwd Connection Test - Local SN	Workflow for local ServiceNow instance connection test.
Pwd Connection Test - Master	A master workflow to test credential store connectivity.
Pwd Connection Test - Mock Failure	An example credential store connection test that simulates a failed connection.
Pwd Connection Test - Mock Success	An example credential store connection test that simulates a successful connection.

### Get User Unlock State Workflows

The following workflows are used to get a user's unlock state. These workflows are available starting with the Eureka release.

Workflow	Description
Pwd Get Lock State - Local SN	Workflow to get a user's lock state for local ServiceNow instance.
Pwd Get Lock State - Master	A master workflow to get a user's lock state.

Unlock User Workflows

The following workflows are used to unlock a user. These workflows are available starting with the Eureka release.

Workflow	Description
Pwd Unlock Account - Local SN	Workflow to unlock a user's account for a local ServiceNow instance.
Pwd Unlock Account - Master	A master workflow to unlock a user's account.

Creating Credential Store Types

- 1. Navigate to **Password Reset > Credential Store Types**.
- 2. Click **New**.
- 3. Fill in the fields, as appropriate (see table).
- 4. Click **Submit**.

<

≡

Password Reset Credential Store Types - Local ServiceNow Instance

Update

Delete

🔗

⬆

⬇

Name

Local ServiceNow Instance

Description

Represents the current (local) ServiceNow instance

Password reset workflow

Pwd Reset - Local Service

Get user lock state workflow

Pwd Get Lock State - Loc

Connection test workflow

Pwd Connection Test - Lc

Unlock user workflow

Pwd Unlock Account - Lo

Update

Delete

≡

Password Reset Credential Stores

New

Go to

Name

🔍

<<

<

1

to 1 of 1

>

>>

Type = Local ServiceNow Instance

🔍

≡

Name

≡

Hostname

≡

Description

Local ServiceNow Instance

N/A

This is a credential store entry for the...

Actions on selected rows...

<<

<

1

to 1 of 1

>

>>

New credential store type

Field	Description
Name	Unique name of the credential store type.
Description	Description of the credential store type.
Password reset workflow	Subflow that defines the credential store processing. Password reset workflows are available to use as a model. You need to provide scripts for each of the activities defined for the subflow. For more on how to create a workflow, see Creating a Workflow.
Connection test workflow	[Optional] Subflow that defines how to test the connection. Connection test workflows are available to use as a model. If you create a connection test subflow, you need to provide scripts for each of the activities defined for the subflow.
Get user lock state workflow	[Optional] Subflow that defines how to get the user lock state. Get user lock state workflows are available to use as a model. You need to provide scripts for each of the activities defined for the subflow. For more on how to create a workflow, see Creating a Workflow. This field is available starting with the Eureka release.
Unlock user workflow	[Optional] Subflow that defines how to unlock a user. Unlock user workflows are available to use as a model. If you create a connection test subflow, you need to provide scripts for each of the activities defined for the subflow. This field is available starting with the Eureka release.

# Defining Verification Types



**Note:** This article applies to Fuji. For more current information, see *Password Reset Verifications* <sup>[1]</sup> at <http://docs.servicenow.com> The ServiceNow Wiki is no longer being updated. Please refer to <http://docs.servicenow.com> for the latest product documentation.

## Overview

A verification type is a set of processors and UI macros that specify how a user is to be enrolled for and have their identity verified for password reset. ServiceNow installs several verification types, and you can create new ones. For information on how to use verification types, see *Configuring Verifications*.

## Anatomy of a Verification Type

Users can be automatically enrolled in password reset, or you can define an enrollment check. To define an enrollment check, you must define both an enrollment processor script, which specifies how enrollment is processed, and a corresponding enrollment UI macro, which specifies how the enrollment information is displayed to the user. For example, the following verification type uses *PwdEnrollSMSProcessor* as the enrollment processor script and *pwd\_enroll\_sms\_ui* as the enrollment UI macro.

Password Reset Verification Type

UpdateDelete

Name:

SMS Code Verification Type

Description:

Verify user's identity by sending them an SMS code and verifying that the code matches

Enrollment check:

PwdSMSEnrollmentCheck

Automatic enrollment:

☐

Enrollment UI:

pwd\_enroll\_sms\_ui

Enrollment processor:

PwdEnrollSMSProcessor

Verification UI:

pwd\_verify\_sms\_ui

Verification processor:

PwdVerifySMSProcessor

UpdateDelete

Password Reset Verifications

New

Go to

Order

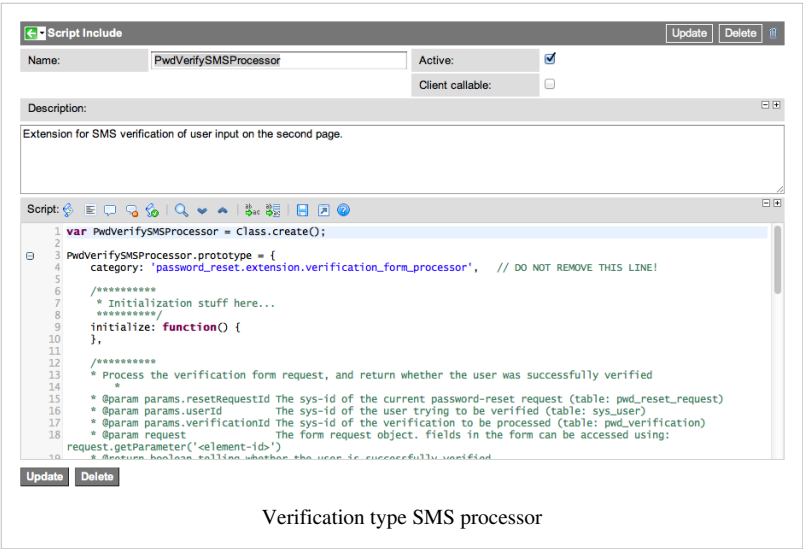
1 to 1 of 1

Type = SMS Code Verification Type

Name	Description	Order
SMS Verification		100

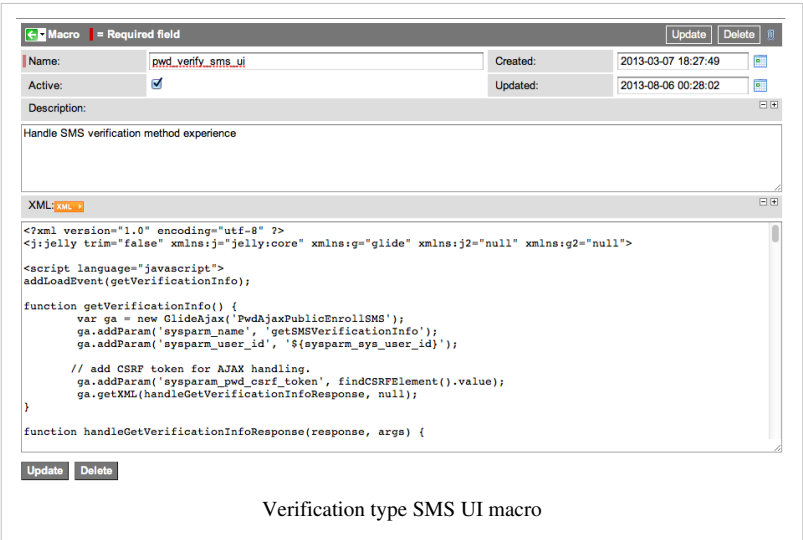
Actions on selected rows...

A verification type must also define a verification processor script and a corresponding verification UI macro. The example uses *PwdVerifySMSProcessor* as the verification processor script and *pwd\_verify\_sms\_ui* as the verification UI macro. The following shows the verification processor script:



Verification type SMS processor

The following is the corresponding UI macro:



Verification type SMS UI macro

# Installed Verification Types

ServiceNow installs these verification types, which you can use as-is or as a model to create custom verification types.

Name	Description
Mock Verification Type	Demonstrates how to add a new verification type in a password reset process, and is not an actual identity verification. This example gets input from the user and returns the entered value through the verification processor. The enrollment UI macro for this verification gets the user input and returns the entered value in the enrollment processor.
Personal Data Confirmation Verification Type	Verifies a user's identity by using data from the User [sys_user] table. The user's data is displayed to a service desk employee who verifies the data. Enrollment is not required for this type. (Recommended for service desk processes.)
Personal Data Verification Type	Verifies a user's identity by using data from the User table. The user is required to answer a question. Enrollment is not required for this type.
Security Questions Verification Type	Verifies a user's identity by answering personal security questions.
SMS Code Verification Type	Verifies a user's identity by sending an SMS code and verifying that the code matches.

# Creating Verification Types

- 1. Navigate to **Password Reset > Verification Types**.
- 2. Click **New**.
- 3. Fill in the fields, as appropriate (see table).
- 4. Click **Submit**.

Password Reset Verification Type

Required field

Submit

Name:

Description:

Enrollment check: -- None --

Automatic enrollment:

Enrollment UI:

Enrollment processor: -- None --

Verification UI:

Verification processor: -- None --

Submit

New verification type

Field	Description
Name	Unique name of the verification type.
Description	Description of the verification type.
Enrollment check	Script to check whether a user is enrolled for verification. For a description of the default enrollment check script includes, see Enrollment Check Script Includes. To define a new script, see Creating Extension Scripts. Automatic enrollment requires an enrollment check. If no script is specified when <b>Automatic Enrollment</b> is selected, a default script is provided automatically.
Automatic enrollment	Indicator for whether users are automatically enrolled. If this is not selected, an enrollment UI macro and enrollment processor script must be provided.
Enrollment UI	Enrollment UI macro that provides the UI for the enrollment. For a description of the predefined password reset UI macros, see Password Reset UI Macros. To define a new macro, see UI Macros.
Enrollment processor	Enrollment processor script that processes the enrollment. For a description of the predefined enrollment processor script includes, see Enrollment Processor Script Includes. To define a new script, see Creating Extension Scripts.
Verification UI	Verification UI macro that provides the UI for the verification. For a description of the predefined password reset UI macros, see Password Reset UI Macros. To define a new macro, see UI Macros.
Verification processor	Verification processor script that processes the verification. For a description of the predefined verification processor script includes, see Verification Processor Script Includes. To define a new script, see Creating Extension Scripts.

## References

[1] [https://docs.servicenow.com/bundle/jakarta-it-operations-management/page/administer/login/concept/c\\_PWRVerifications.html](https://docs.servicenow.com/bundle/jakarta-it-operations-management/page/administer/login/concept/c_PWRVerifications.html)

# Defining Identification Types

## Overview

Identification types allow you to override the default verification methods associated with a password reset process, allowing users to enter alternative verification information such as an employee ID to verify their identity. For more information on how an identification type is used to configure password reset, see [Configuring Processes](#).

## Identification Types

ServiceNow installs these identification types, which you can use as-is or as a model to create custom identification types.

Name	Description
Email Identification	Identifies users by their email addresses.
Username Identification	Identifies users by their usernames.

## Creating Identification Types

- 1. Navigate to **Password Reset > Identification Types**.
- 2. Click **New**.
- 3. Fill in the fields, as appropriate (see table).
- 4. Click **Submit**.

New identification type

Field	Description
Name	Unique name of the identification type.
Description	Description of the identification type.
Identification field label	Text to display as a label for the Identification field.
Identification processor	Identification processor script. Choose an existing script, or create your own using the Identification form processor category, as described in <a href="#">Creating Extension Scripts</a> .



# Creating Extension Scripts

## Overview

Extension scripts allow you to extend password reset functionality by defining custom scripts that can be used in credential store, verification, or identification types. Extension scripts are predefined hooks within the Password Reset application that perform specific types of behavior defined by the extension *category*, which refers to where the script can be used. This page describes how to create, edit, and use extension scripts. For detailed reference information, see Password Reset Extension Script Includes.

ServiceNow installs several scripts in each category. You can use them as-is or as a template for creating custom scripts. For a description of these scripts, see Installed Password Reset Scripts.

## Where To Use Extension Scripts

The category indicates where a script can be used in the Password Reset application. For example, a script in the **Enrollment check** category can be selected to perform the enrollment check for a verification.

Category	Description
Enrollment check	Defines how enrollment is to be checked. Scripts of this category are available in the <b>Enrollment check</b> field when you define a verification type (Password Reset Verification Type form). See the Enrollment Check script include.
Enrollment form processor	Defines how an enrollment form is processed (if not automatic enrollment). Scripts of this category are available in the <b>Enrollment form processor</b> field when you define a verification type (Password Reset Verification Type form). See the Enrollment Form Processor script include.
Identification form processor	Defines how an identification is processed. Scripts of this category are available in the <b>Identification processor</b> field when you define a verification type (Password Reset Identification Type form). See the Identification Form Processor script include.
Password generator	Defines how to automatically generate a password. Scripts of this category are available in the <b>Auto generate password</b> field when you configure a credential store (Password Reset Credential Store Type form). See the Password Generator script include.
Post reset script	Executes at the end of a password reset process. Scripts of this category are available in the <b>Post reset script</b> field when you configure a process (Password Reset Process form). See the Password Reset script include.
User account lookup	Defines how a user account lookup is performed. Scripts of this category are available in the <b>User account lookup</b> field when you configure a credential store (Password Reset Credential Store form). See the User Account Lookup script include.
Verification form processor	Defines how a verification form is processed. Scripts of this category are available in the <b>Verification processor</b> field when you define a verification type (Password Reset Verification Type form). See the Verification Form Processor script include.

## Creating Extension Scripts



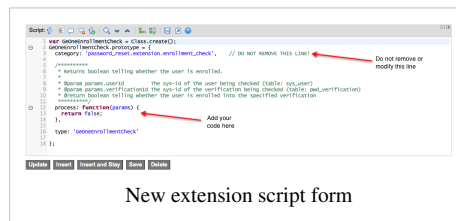
**Note:** You should create new extension scripts only from the Password Reset Extension Script form (**Password Reset > Extensions > New extension script**) as described here. Extension scripts are special purpose script includes that should not be created in the **System Definition > Script Includes** interface.

1. Go to **Password Reset > Extensions > New extension script**.
2. Enter a name in the **Extension script name** field.
3. Select an **Extension script category** (see table). For details on the script include for each category, see Extension Script Includes.
  - Enrollment check

- Enrollment form processor
  - Identification form processor
  - Password generator
  - User account lookup
  - Verification form processor
4. Click **Submit** to open a new instance of the script include for the selected category.
  5. Edit the script by providing an implementation in the body of the process or processForm function, depending on the script category. You can add additional functions as long as the process or processForm function accepts the defined parameters and returns the expected results.



**Note:** Do not edit or delete the Category declaration.



## Example

The following example shows a script that uses two sample extensions, one process extension and one processForm extension.

```
// User account lookup
var lookupExtensionSysId =
getExtensionScriptSysId('SampleUserAccountLookupExtension',
'user_account_lookup');
var lookupExtension = new SNC.PwdExtensionPoint(lookupExtensionSysId);

// Setup parameters required for this extension type - userId
var params = new SNC.PwdExtensionPointParameter();
params.userId= 'joe.employee';

// Invoke the extension
var answer = lookupExtension.process(params);
gs.print('user: ' + answer);

//Form processor sample - Identification form processor
var identExtensionSysId =
getExtensionScriptSysId('SampleIdentificationProcessorExtension',
'identification_form_processor');
var identificationExtension = new
SNC.PwdExtensionPoint(identExtensionSysId);

// Setup parameters required for this extension type - processId
```

```

var params = new SNC.PwdExtensionPointParameter() ;
params.processId = 'pwdreq1234';

// Simulate the posted form parameter for the identification processor
var request = new SNC.PwdExtensionPointParameter() ; // A real life
case will inject it's own request object
request.setParameter('sysparm_user_id', 'joe.employee');

var userIdentity = identificationExtension.processForm(params,
request);
gs.print('identity: ' + userIdentity);

// Simple helper to return the sys-id for a given extension script
function getExtensionScriptSysId(scriptName, category) {
    var result;
    var gr = new GlideRecord('sys_script_include');
    gr.addQuery('name', scriptName);
    gr.addQuery('script', 'CONTAINS', 'category:
\'password_reset.extension.' + category + '\');
    gr.query();

    if (gr.next() ) {
        result = gr.getValue('sys_id');
    }
    return result;
}

```

process function:

```

<?xml version="1.0" encoding="UTF-8"?>
<record_update table="sys_script_include">
  <sys_script_include action="INSERT_OR_UPDATE">
    <active>true</active>
    <client_callable>false</client_callable>
    <description>Simple account lookup that returns the supplied user id</description>
    <name>SampleUserAccountLookupExtension</name>
    <script><![CDATA[
var SampleUserAccountLookupExtension = Class.create();
SampleUserAccountLookupExtension.prototype = {
  category: 'password_reset.extension.user_account_lookup',      // DO
NOT REMOVE THIS LINE!

  /*****
   * Returns the credential-store account id for a given user
   *
   * @param params.userId The sys-id of the user being checked (table:
sys_user)
   * @return              The credential-store account-id (string) for
a given user

```

```

*****/

process: function(params) {
    return params.userId;
},

type: 'SampleUserAccountLookupExtension'

];]]</script>

<sys_created_by>admin</sys_created_by>
<sys_created_on>2013-07-30 16:44:55</sys_created_on>
<sys_id>2df5a103d73201002bb9af728e610333</sys_id>
<sys_mod_count>1</sys_mod_count>
<sys_updated_by>admin</sys_updated_by>
<sys_updated_on>2013-07-30 16:46:00</sys_updated_on>
</sys_script_include>
<sys_app_file action="INSERT_OR_UPDATE">
    <customer_update>false</customer_update>
    <publish_override/>
    <replace_on_upgrade>false</replace_on_upgrade>
    <restore/>
    <sys_app/>
    <sys_code>!!1W4</sys_code>
    <sys_created_by>admin</sys_created_by>
    <sys_created_on>2013-07-30 16:44:55</sys_created_on>
    <sys_id>8306e143d73201002bb9af728e6103d3</sys_id>
    <sys_mod_count>0</sys_mod_count>
    <sys_name>SampleUserAccountLookupExtension</sys_name>
    <sys_parent/>
    <sys_path>!!1W4</sys_path>
    <sys_policy/>
    <sys_source_deleted>false</sys_source_deleted>
    <sys_source_id>2df5a103d73201002bb9af728e610333</sys_source_id>
    <sys_source_table>sys_script_include</sys_source_table>
    <sys_type>code</sys_type>
    <sys_update_name>sys_script_include_2df5a103d73201002bb9af728e610333</sys_update_name>
    <sys_updated_by>admin</sys_updated_by>
    <sys_updated_on>2013-07-30 16:46:00</sys_updated_on>
</sys_app_file>
</record_update>

```

processForm function:

```

<?xml version="1.0" encoding="UTF-8"?>
<record_update table="sys_script_include">
    <sys_script_include action="INSERT_OR_UPDATE">
        <active>true</active>
        <client_callable>false</client_callable>
        <description>Script that processes an identification form.&#13;

```

```

Returns the sys-id of the user that corresponds to the requested input;
  if no user was found, null should be returned.&#13;
</description>
  <name>SampleIdentificationProcessorExtension</name>
  <script><![CDATA[var SampleIdentificationProcessorExtension = Class.create();
SampleIdentificationProcessorExtension.prototype = {
  category: 'password_reset.extension.identification_form_processor',
    // DO NOT REMOVE THIS LINE!

  /*****
   * Process the identification form request, and returns the user's
   sys_id. if user was not identified return null.
   *
   * @param params.processId The sys-id of the calling password-reset
   process (table: pwd_process)
   * @param request The form request object. fields in the
   form can be accessed using: request.getParameter('<element-id>')
   *
   Supported request paramters:
   *
   sysparm_user_id - the user
   identifier value entered in the form.
   * @return The sys-id of the user that corresponds to the requested
   input; if no user was found, null should be returned.
   *****/
  processForm: function(params, request) {
    return request.getParameter('sysparm_user_id') + '_' +
params.processId;
  },

  type: 'SampleIdentificationProcessorExtension'

};]]></script>
  <sys_created_by>admin</sys_created_by>
  <sys_created_on>2013-07-30 17:00:28</sys_created_on>
  <sys_id>3a79a503d73201002bb9af728e610349</sys_id>
  <sys_mod_count>1</sys_mod_count>
  <sys_updated_by>admin</sys_updated_by>
  <sys_updated_on>2013-07-30 17:08:41</sys_updated_on>
</sys_script_include>
<sys_app_file action="INSERT_OR_UPDATE">
  <customer_update>false</customer_update>
  <publish_override/>
  <replace_on_upgrade>false</replace_on_upgrade>
  <restore/>
  <sys_app/>
  <sys_code>!!1W5</sys_code>
  <sys_created_by>admin</sys_created_by>
  <sys_created_on>2013-07-30 17:00:28</sys_created_on>

```

```

<sys_id>4799ed03d73201002bb9af728e610333</sys_id>
<sys_mod_count>0</sys_mod_count>
<sys_name>SampleIdentificationProcessorExtension</sys_name>
<sys_parent/>
<sys_path>!!lW5</sys_path>
<sys_policy/>
<sys_source_deleted>false</sys_source_deleted>
<sys_source_id>3a79a503d73201002bb9af728e610349</sys_source_id>
<sys_source_table>sys_script_include</sys_source_table>
<sys_type>code</sys_type>
<sys_update_name>sys_script_include_3a79a503d73201002bb9af728e610349</sys_update_name>
<sys_updated_by>admin</sys_updated_by>
<sys_updated_on>2013-07-30 17:08:41</sys_updated_on>
</sys_app_file>
</record_update>

```

## Extension Script Includes

---

### Overview

Password reset scripts allow you to extend password reset functionality by creating your own credential store, verification, and identification types, and by adding operations to password reset processes. A script include is associated with a specific category, which is available in the appropriate field of a password reset form. For information on how to use the password reset script includes, see [Creating Extension Scripts](#).

ServiceNow installs several scripts in each category. You can use them as-is or as a template for creating custom scripts. For a description of these scripts, see [Installed Password Reset Scripts](#).



**Note:** You should create new extension scripts only from the Password Reset Extension Script form (**Password Reset > Extensions > New extension script**). Extension scripts are special purpose script includes that should not be created in the **System Definition > Script Includes** interface.

### Where to Use

Use these script includes for customizing a password reset process.

### Script Includes Summary

Name	Description
Enrollment check	Checks if the user is enrolled for a given verification.
Enrollment form processor	Checks if all necessary information has been collected from the user and stores the information so it can be used for verification when the user resets his or her password.
Identification form processor	Processes an identification form.
Password generator	Gets an automatically generated password.
Post reset script	Performs additional operations after the completion of the password reset process.
User account lookup	Gets the credential store account ID for a given user.
Verification form processor	Processes a verification form.

## Script Include Details

### Enrollment Check

Checks if a user is enrolled for a given verification.

### Method Signature

```
process(params)
```

### Input Fields

#### Parameters:

- **params.userId** - The sys\_id of the user to check (table: sys\_user).
- **params.verificationId** - The sys\_id of the verification to check (table: pwd\_verification).

### Output Fields

**Returns:** (boolean) true, if the user is enrolled in the specified verification; otherwise, false.

### Example

This example signals that the user is enrolled if both expected parameters are supplied. The code would be contained in the **Script** field of an extension script named SampleEnrollmentCheck:

```
var SampleEnrollmentCheck = Class.create();
SampleEnrollmentCheck.prototype = {
  category: 'password_reset.extension.enrollment_check', // DO NOT
  REMOVE THIS LINE!

  /*****
   * Returns boolean telling whether the user is enrolled.
   * This sample returns true if both parameters are supplied, false
   otherwise
   *
   * @param params.userId      The sys-id of the user being checked
   (table: sys_user)
```

```

    * @param params.verificationId The sys-id of the verification being
checked (table: pwd_verification)
    * @return Boolean indicating whether the user is enrolled into the
specified verification
    *****/
process: function(params) {
    return (params.userId && params.verificationId) ? true : false;
},

type: 'SampleEnrollmentCheck'
};

```

## Enrollment Form Processor

Checks if all necessary information has been collected from the user. Stores the information so it can be used for verification when the user resets his or her password.

### Method Signature

```
process(params)
```

### Input Fields

#### Parameters:

- **params.resetRequestId** - The sys\_id of the current password reset request (table: pwd\_reset\_request).
- **params.userId** - The sys\_id of the user to be verified (table: sys\_user).
- **params.verificationId** - The sys\_id of the verification to be processed (table: pwd\_verification).
- **request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`.

The following information should be added to the state of the enrollment process:

- **gs.getSession().putProperty("result.status",status)** - Whether the user was successfully enrolled.
- **gs.getSession().putProperty("result.message",message)** - An associated message to be returned to the UI, such as a detailed error message.
- **gs.getSession().putProperty("result.value",value)** - A custom value associated with the enrollment.

### Output Fields

**Returns:** (boolean) true, if the user is enrolled in the specified verification; otherwise, false.

### Example

This example processes an enrollment form submission successfully if the user-submitted response was success. The code would be contained in the **Script** field of an extension script named `SampleEnrollmentProcessor`:

```

var SampleEnrollmentProcessor = Class.create();
SampleEnrollmentProcessor.prototype = {
    category: 'password_reset.extension.enrollment_form_processor',
    // DO NOT REMOVE THIS LINE!

    /*****
    * Process the enrollment form request, and return whether the user

```



```

was successfully enrolled.
*
* @param params.userId          The sys_id of the user trying to
enroll (table: sys_user)
* @param params.verificationId The sys_id of the verification to be
enrolled into (table: pwd_verification)
* @param params.enrollmentId    The sys_id of this enrollment
process
* @param request                The form request object. Fields in
the form can be accessed with
*
request.getParameter('<element-id>')
* @return boolean telling whether the user was successfully
enrolled
* The following information should be added to the state of the
enrollment process
*     gs.getSession().putProperty("result.status",status) - whether
the user was successfully enrolled
*     gs.getSession().putProperty("result.message",message) - an
associated message to be returned
*         to the UI. Eg. a detailed error message
*     gs.getSession().putProperty("result.value",value) - custom
value associated with the enrollment
*****/
processForm: function(params, request) {
    var verificationId = params.verificationId;
    var sampleInput = request.getParameter('sample_input');

    if (gs.nil(verificationId) || (sampleInput != 'success')) {
        return false;
    }

    var gr = new GlideRecord('sys_user');
    gr.get(params.userId);
    gs.print('User: ' + gr.getValue('user_name') + ' successfully
enrolled');
    return true;
},
type: 'SampleEnrollmentProcessor'
};

```

## Identification Form Processor

Processes an identification form request.

### Method Signature

```
processForm(params, request)
```

### Input Fields

#### Parameters:

- **params.processId** - The sys\_id of the calling password reset process (table: pwd\_process).
- **request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`. Use `request.getParameter('sysparm_user_id')` to get the user ID that was entered into the form.

### Output Fields

**Returns:** the sys\_id of the user that corresponds to the requested input; if no user was found, returns null.

### Example

This example attempts to identify the user within the sys\_user table given a user name submitted from the identification form. The code would be contained in the **Script** field of an extension script named PwdIdentifyViaUsername:

```
var PwdIdentifyViaUsername = Class.create();
PwdIdentifyViaUsername.prototype = {
  category: 'password_reset.extension.identification_form_processor',
  // DO NOT REMOVE THIS LINE!

  initialize: function() {
  },

  /*****
   * Process the identification form request, and returns the user's
   sys_id. If user was not identified return null.
   *
   * @param params.processId The sys_id of the calling password reset
   process (table: pwd_process)
   * @param request The form request object. fields in the
   form can be accessed with
   * request.getParameter('<element-id>')
   * Supported request parameters: sysparm_user_id - the user
   identifier value entered in the form
   * @return The sys_id of the user that corresponds to the requested
   input;
   * if no user was found, null should be returned
   *****/
  processForm: function(params, request) {
    var processId = params.processId;
```

```

    var sysparm_user_id = request.getParameter('sysparm_user_id');
    gr = new GlideRecord('sys_user');
    gr.addQuery('user_name', sysparm_user_id);
    gr.query();
    if (!gr.next()) {
        return null;
    }
    return gr.sys_id;
},

type: 'PwdIdentifyViaUsername'
}

```

## Password Generator

Returns an auto-generated password.

### Method Signature

process(params)

### Input Fields

#### Parameters:

- **params.processId** - The sys\_id of the calling password reset process (table: pwd\_process).

### Output Fields

**Returns:** (String) an auto-generated password.

### Example

This example randomly generates a password from a base word and numbers. The base word is selected depending on the credential store. The code would be contained in the **Script** field of an extension script named SamplePasswordGenerator:

```

var SamplePasswordGenerator = Class.create();
SamplePasswordGenerator.prototype = {
    category: 'password_reset.extension.password_generator', // DO
    NOT REMOVE THIS LINE!

    /**
     * Returns an auto-generated string password.
     * This sample randomly generates 4 digits to add to the password.
     *
     * @param params.credentialStoreId The sys_id of the target password
    reset credential store to generate
     * a password for (table: pwd_cred_store)
     * @return An auto-generated string password
     *****/
    process: function(params) {

```

```

    var basePassword;

    var gr = new GlideRecord('pwd_cred_store');
    gr.addQuery('name', 'Local ServiceNow Instance');
    gr.query();
    if (gr.next()) {
        if (params.credentialStoreId == gr.getValue('sys_id'))
            basePassword = "Password";
        else
            basePassword = "Dorwssap";
    }
    return this.generateSimple(basePassword);
},

generateSimple : function(base) {
    var pwd = base;
    var numbers = '0123456789';
    var length = 4;

    for (var i = 0, n = numbers.length; i < length; i++) {
        pwd += numbers.charAt(Math.floor(Math.random() * n) + 1);
    }
    return pwd;
},

type: 'SamplePasswordGenerator'
};

```

## Post Reset

Performs additional operations after the completion of the password reset process.

## Method Signature

process(params)

## Input Fields

### Parameters:

- **params.resetRequestId** - The sys\_id of the calling password reset process (table: pwd\_process).
- **params.wfSuccess** - A flag indicating whether the workflow completed successfully. True if, and only if, successful.

## Output Fields

**Returns:** void

## Example

This example adds failed reset requests to the system log. The code would be contained in the **Script** field for an extension script named `PwdPostProcessor`:

```
var PwdPostProcessor = Class.create();

PwdPostProcessor.prototype = {
  category: 'password_reset.extension.post_reset_script',      //
  DO NOT REMOVE THIS LINE!

  initialize: function() {
  },

  /*****
   * Execute custom actions after the password reset process has
   completed.
   *
   * @param params.resetRequestId The sys_id of the current password
   reset request (table: pwd_reset_request)
   * @param params.wfSuccess      A flag indicating if the workflow
   completed successfully.
   * True if (and only if) successful.
   * @return no return value
   *****/
  process: function(params) {
    if (!params.wfSuccess) {
      gs.log('[PwdPostProcessor.process] failure post processing
for request [' + params.resetRequestId + ']');
    }

    // We could place actions here that we always want executed
    return;
  },

  type: 'PwdPostProcessor'
}
```

## User Account Lookup

Gets the credential store account ID for a given user.

### Method Signature

process(params)

### Input Fields

#### Parameters:

- **params.userId** - The sys\_id of the user being checked (table: sys\_user).

### Output Fields

**Returns:** (String) the credential store account ID for the given user.

### Example

This example gets the credential store account for a user. This code would be contained in the **Script** field of an extension script named SampleUserAccountLookupExtension:

```
var SampleUserAccountLookupExtension = Class.create();
SampleUserAccountLookupExtension.prototype = {
  category: 'password_reset.extension.user_account_lookup',      // DO
  NOT REMOVE THIS LINE!

  /*****
   * Returns the credential store account id for a given user.
   * This sample simply echoes the user_id supplied as the credential
   store account id for that user.
   *
   * @param params.userId The sys_id of the user being checked (table:
sys_user)
   * @return The credential store account id (string) for
a given user
   *****/
  process: function(params) {
    return params.userId;
  },

  type: 'SampleUserAccountLookupExtension'
}
```

## Verification Form Processor

Processes a verification form request and indicates whether the user was verified or not.

### Method Signature

```
processForm(params, request)
```

### Input Fields

#### Parameters:

- **params.resetRequestId** - The sys\_id of the current password reset request (table: pwd\_reset\_request).
- **params.userId** - The sys\_id of the user to be verified (table: sys\_user).
- **params.verificationId** - The sys\_id of the verification to be processed (table: pwd\_verification).
- **request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`.

### Output Fields

**Returns:** (boolean) true, if the user is verified; otherwise, false.

### Example

This example shows a verification processor that returns true only if the user sent **ok** in the input field; otherwise, it returns false. The code would be contained in the **Script** field of an extension script named `SampleVerificationFormProcessor`:

```
var SampleVerificationFormProcessor = Class.create();
SampleVerificationFormProcessor.prototype = {
  category: 'password_reset.extension.verification_form_processor',
    // DO NOT REMOVE THIS LINE!

  /**
   * Process the verification form request, and return whether the user
   * was successfully verified.
   * This is a sample verification processor returns true only if the
   * user sent "ok" in the input field;
   * otherwise, it returns false.
   *
   * @param params.resetRequestId The sys_id of the current password
   * reset request (table: pwd_reset_request)
   * @param params.userId The sys_id of the user trying to be
   * verified (table: sys_user)
   * @param params.verificationId The sys_id of the verification to be
   * processed (table: pwd_verification)
   * @param request The form request object. Fields in
   * the form can be accessed with
   * request.getParameter('<element-id>')
   * @return Boolean indicating whether the user is successfully
   * verified
   */
  processForm: function(params, request) {
    // *****/
  }
};
```

```
processForm: function(params, request) {
  if (request.getParameter("sysparm_simple_input") == "ok")
    return true;
  else
    return false;
},

type: 'SampleVerificationFormProcessor'
};
```

## Installed Scripts

---

### Overview

ServiceNow installs several password reset scripts, which you can use as-is or customize. For information on how to customize password reset script includes, see [Creating Extension Scripts](#). When you create a new extension, it must take the input values and return the values described on this page for the extension category.



**Note:** You should create new extension scripts only from the Password Reset Extension Script form (**Password Reset > Extensions > New extension script**). Extension scripts are special purpose script includes that should not be created in the **System Definition > Script Includes** interface.

### Where to Use

Use these script includes as-is or as a basis for creating customized password reset extension scripts.

### Enrollment Check Script Includes

Enrollment check script includes provide functionality for extending enrollment checks. All enrollment check script includes take the following parameters, and return a boolean indicating whether the user is verified:

- **params.userId** - The sys\_id of the user being checked (table: sys\_user).
- **params.verificationId** - The sys\_id of the verification being checked (table: pwd\_verification).

Name	Description
PwdAlwaysEnrolled	Provides a default check that always returns true.
PwdMockIsEnrolled	Provides an example check that always returns true.
PwdQuestionsEnrollmentCheck	Determines if a user has enrolled for password reset using security question verification.
PwdSMSEnrollmentCheck	Determines if a user has enrolled for password reset using SMS verification.



## Identification Form Processor Script Includes

Identification form processor script includes provide functionality for extending identification processing. All identification form processor script includes take the following parameters, and return the `sys_id` of the user that corresponds to the requested input, or if the user was not identified, return null.

- **params.processId** - The `sys_id` of the calling password reset process (table: `pwd_process`).
- **param request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`. The supported request parameter is `sysparm_user_id`, the user identifier value entered in the form.

Name	Description
<code>PwdIdentifyViaEmail</code>	Verifies a user's identity by checking his or her email address.
<code>PwdIdentifyViaUsername</code>	Verifies a user's identity by checking his or her user name.

## Enrollment Form Processor Script Includes

Enrollment form processor script includes (see table) provide functionality for extending enrollment form processing. All enrollment form processor script includes take the following parameters, and return a boolean indicating whether the user was successfully enrolled.

- **params.userId** - The `sys_id` of the user trying to enroll (table: `sys_user`).
- **params.verificationId** - The `sys_id` of the verification used to enroll (table: `pwd_verification`).
- **params.enrollmentId** - The `sys_id` of this enrollment process.
- **request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`.

The following information should be added to the state of the enrollment process:

- **gs.getSession().putProperty("result.status",status)** - Whether the user was successfully enrolled.
- **gs.getSession().putProperty("result.message",message)** - An associated message to be returned to the UI, such as a detailed error message.
- **gs.getSession().putProperty("result.value",value)** - A custom value associated with the enrollment.

Name	Description
<code>PwdEnrollQuestionsProcessor</code>	Handles questions and answers for verification.
<code>PwdEnrollSampleProcessor</code>	Provides an enrollment processor for sample verification.
<code>PwdEnrollSMSProcessor</code>	Provides an enrollment processor for SMS verification.

## User Account Lookup Script Includes

User account lookup script includes take the following parameter and return the credential store `account_id` for a given user.

- **params.userId** - The `sys_id` of the user being checked (table: `sys_user`).

Name	Description
<code>PwdDefaultUserAccountLookup</code>	Provides a default script for user account lookup from a <code>user_id</code> to the account in a credential store. The default mapping is to use the user name as the account name.

## Password Generator Script Includes

Password generator script includes take the following parameter, and return an auto-generated string password.

- **params.credentialStoreId** - The sys\_id of the calling password reset process (table: pwd\_process).

Name	Description
PwdDefaultAutoGenPassword	Generates a password from a random word and 4 digits.

## Verification Processor Script Includes

Verification processor script includes take the following parameters, and return true if the user is verified.

- **params.resetRequestId** - The sys\_id of the current password reset request (table: pwd\_reset\_request).
- **params.userId** - The sys\_id of the user to be verified (table: sys\_user).
- **params.verificationId** - The sys\_id of the verification (table: pwd\_verification).
- **request** - The form request object. Fields in the form can be accessed with `request.getParameter('<element-id>')`.

Name	Description
PwdVerifyPersonalDataConfirmationProcess	Verifies that the answer was accepted by the user.
PwdVerifyPersonalDataProcessor	Verifies that the user's answers match the expected data in the system.
PwdVerifyQuestionsProcessor	Provides question and answer verification of user input on the second page of the verification form.
PwdVerifySimpleProcessor	Provides simple verification of user input on the second page of the verification form.
PwdVerifySMSProcessor	Provides SMS verification of user input on the second page of the verification form.

## Post Processor Script Includes

Post processor script includes execute custom actions after the password reset process has completed. All post processor script includes take the following parameters.

- **params.resetRequestId** - The sys\_id of the current password reset request (table: pwd\_reset\_request).
- **params.wfSuccess** - A flag indicating whether the workflow completed successfully. True if, and only if, successful.

Name	Description
PwdPostProcessor	Executes actions after the process completes for success, failure, or both conditions.

# UI Macros

---

## Overview

ServiceNow installs several password reset UI macros, which you can use as-is or as a template for creating custom macros. For information on how to customize macros, see [UI Macros](#). ServiceNow also installs several password reset UI Scripts, which can be referenced from a UI macro.

## Where to Use

Use password reset UI macros and scripts when defining a UI for a custom enrollment or verification type.

## UI Macros

The following table describes the installed password reset macros.

Name	Description
pwd_enrollment_form_title	A Jelly macro function that prints the title for the enrollment form. A verification ID is mandatory.
pwd_enroll_questions_ui	A UI for question and answer security validation enrollment.
pwd_enroll_questions_ui_js	JavaScript code that requires server-side data for security question and answer enrollment.
pwd_enroll_sample_ui	A sample UI macro for enrollment for Mock Verification Type.
pwd_enroll_sms_ui	A UI for SMS verification enrollment.
pwd_verify_personal_data_confirmation_ui	A UI for verifying personal data confirmation.
pwd_verify_personal_data_ui	A UI for verifying personal data.
pwd_verify_questions_ui	A UI for verifying questions.
pwd_verify_simple_ui	An input section for a simple verification method. This is a single input field.
pwd_verify_sms_ui	A UI for SMS verification.

## UI Scripts

You can create a UI script and reference the script from a UI macro or UI page by using a `<g:include_script>` Jelly tag. For example, the following shows how the `pwd_enroll_questions_ui` script can be referenced by the `pwd_enroll_questions_ui` UI macro, where `[UI Script Name]+".jsdbx"` is the name of the script:

```
<g:include_script src="pwd_enroll_questions_ui.jsdbx" />
```

By referencing an external script, you can maintain separation between client JavaScript code and Jelly code, simplifying maintenance. The following are installed scripts that you can use with password reset UI macros.

---

Name	Description
pwdWfManager	A helper class to handle workflow activities and post-processing.
pwd_csrf_common_ui_script	A common UI script for handling a Cross-site Request Forgery (CSRF).
pwd_enrollment_submit_event	A UI script for an enrollment submit event.
pwd_enroll_questions_ui	JavaScript code for the pwd_enroll_questions_ui UI macro.
pwd_enroll_sample_ui	Included sample client JavaScript for the pwd_enroll_sample_ui UI macro.
pwd_enroll_sms_ui	An SMS enrollment UI script.

---

# Article Sources and Contributors

**Password Reset** *Source:* <http://wiki.servicenow.com/index.php?oldid=250771> *Contributors:* Cheryl.dolan, Fuji.publishing.user, George.rawlins, John.ramos, Joseph.messerschmidt, Michael.randall, Rachel.sienko, Roy.lagemann, Vaughn.romero

**Configuring Verifications** *Source:* <http://wiki.servicenow.com/index.php?oldid=245644> *Contributors:* George.rawlins, Michael.randall, Roy.lagemann

**Configuring Credential Stores** *Source:* <http://wiki.servicenow.com/index.php?oldid=250143> *Contributors:* Emily.partridge, George.rawlins, John.ramos, Joseph.messerschmidt, Phillip.salzman

**Configuring Processes** *Source:* <http://wiki.servicenow.com/index.php?oldid=240072> *Contributors:* Fuji.publishing.user, John.ramos, Julie.phaviseth, Phillip.salzman, Roy.lagemann

**Remote Credential Stores** *Source:* <http://wiki.servicenow.com/index.php?oldid=249571> *Contributors:* Cheryl.dolan, George.rawlins, Michael.randall, Roy.lagemann

**Password Reset Desktop Integration (Ctrl+Alt+Del)** *Source:* <http://wiki.servicenow.com/index.php?oldid=249961> *Contributors:* Fuji.publishing.user, Jennifer.ball, Phillip.salzman, Roy.lagemann, Vaughn.romero

**Enrolling** *Source:* <http://wiki.servicenow.com/index.php?oldid=251106> *Contributors:* George.rawlins, John.ramos, Michael.randall

**Resetting Passwords** *Source:* <http://wiki.servicenow.com/index.php?oldid=251163> *Contributors:* Cheryl.dolan, George.rawlins, John.ramos, Joseph.messerschmidt, Roy.lagemann, Vaughn.romero

**Monitoring** *Source:* <http://wiki.servicenow.com/index.php?oldid=241259> *Contributors:* Cheryl.dolan, Emily.partridge, Fuji.publishing.user, George.rawlins, Joseph.messerschmidt, Julie.phaviseth

**Properties** *Source:* <http://wiki.servicenow.com/index.php?oldid=236026> *Contributors:* Cheryl.dolan, George.rawlins, Joseph.messerschmidt, Michael.randall

**CMS Integration** *Source:* <http://wiki.servicenow.com/index.php?oldid=236015> *Contributors:* Cheryl.dolan, George.rawlins

**Installed Components** *Source:* <http://wiki.servicenow.com/index.php?oldid=240977> *Contributors:* Fuji.publishing.user, George.rawlins, Julie.phaviseth, Michael.randall, Vaughn.romero

**Scripting** *Source:* <http://wiki.servicenow.com/index.php?oldid=193274> *Contributors:* Cheryl.dolan, George.rawlins

**Defining Credential Store Types** *Source:* <http://wiki.servicenow.com/index.php?oldid=235349> *Contributors:* Cheryl.dolan, George.rawlins, Joseph.messerschmidt

**Defining Verification Types** *Source:* <http://wiki.servicenow.com/index.php?oldid=250499> *Contributors:* Cheryl.dolan, George.rawlins, John.ramos

**Defining Identification Types** *Source:* <http://wiki.servicenow.com/index.php?oldid=189786> *Contributors:* George.rawlins

**Creating Extension Scripts** *Source:* <http://wiki.servicenow.com/index.php?oldid=189806> *Contributors:* George.rawlins

**Extension Script Includes** *Source:* <http://wiki.servicenow.com/index.php?oldid=189823> *Contributors:* George.rawlins

**Installed Scripts** *Source:* <http://wiki.servicenow.com/index.php?oldid=193273> *Contributors:* Cheryl.dolan, George.rawlins

**UI Macros** *Source:* <http://wiki.servicenow.com/index.php?oldid=189832> *Contributors:* George.rawlins

# Image Sources, Licenses and Contributors

**Image:Warning.gif** *Source:* <http://wiki.servicenow.com/index.php?title=File:Warning.gif> *License:* unknown *Contributors:* CapaJC

**Image:change\_password.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Change\\_password.png](http://wiki.servicenow.com/index.php?title=File:Change_password.png) *License:* unknown *Contributors:* Vaughn.romero

**Image:pwr\_menus\_and\_modules.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwr\\_menus\\_and\\_modules.png](http://wiki.servicenow.com/index.php?title=File:Pwr_menus_and_modules.png) *License:* unknown *Contributors:* Fuji.publishing.user, George.rawlins

**Image:PWDVerificationsForm.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:PWDVerificationsForm.png> *License:* unknown *Contributors:* Michael.randall

**Image:SMSVerificationParams.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:SMSVerificationParams.png> *License:* unknown *Contributors:* Michael.randall

**Image:QAVerificationParams.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:QAVerificationParams.png> *License:* unknown *Contributors:* Michael.randall

**Image:PersonalDataVerificationParams.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:PersonalDataVerificationParams.png> *License:* unknown *Contributors:* Michael.randall

**Image:PWDCredentialStoreForm.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:PWDCredentialStoreForm.png> *License:* unknown *Contributors:* Michael.randall

**Image:password\_reset\_process.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Password\\_reset\\_process.png](http://wiki.servicenow.com/index.php?title=File:Password_reset_process.png) *License:* unknown *Contributors:* Roy.lagemann

**Image:PWDQAVerification.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:PWDQAVerification.png> *License:* unknown *Contributors:* Michael.randall

**Image:PWDSMSEnrollment-YesPreviousDevice.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:PWDSMSEnrollment-YesPreviousDevice.png> *License:* unknown *Contributors:* Michael.randall

**Image:pwr\_reset\_1.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwr\\_reset\\_1.png](http://wiki.servicenow.com/index.php?title=File:Pwr_reset_1.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwr\_reset\_2.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwr\\_reset\\_2.png](http://wiki.servicenow.com/index.php?title=File:Pwr_reset_2.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwr\_reset\_3.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwr\\_reset\\_3.png](http://wiki.servicenow.com/index.php?title=File:Pwr_reset_3.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd cms integration iframe 2.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_cms\\_integration\\_iframe\\_2.png](http://wiki.servicenow.com/index.php?title=File:Pwd_cms_integration_iframe_2.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd cms integration site.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_cms\\_integration\\_site.png](http://wiki.servicenow.com/index.php?title=File:Pwd_cms_integration_site.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd cms integration page.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_cms\\_integration\\_page.png](http://wiki.servicenow.com/index.php?title=File:Pwd_cms_integration_page.png) *License:* unknown *Contributors:* George.rawlins

**Image:Pwd\_Reset\_Local\_ServiceNow\_sub\_workflow.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_Reset\\_Local\\_ServiceNow\\_sub\\_workflow.png](http://wiki.servicenow.com/index.php?title=File:Pwd_Reset_Local_ServiceNow_sub_workflow.png) *License:* unknown *Contributors:* George.rawlins

**Image:Pwd\_Connection\_Test\_Local\_SN\_sub\_workflow.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_Connection\\_Test\\_Local\\_SN\\_sub\\_workflow.png](http://wiki.servicenow.com/index.php?title=File:Pwd_Connection_Test_Local_SN_sub_workflow.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwr credential store types.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwr\\_credential\\_store\\_types.png](http://wiki.servicenow.com/index.php?title=File:Pwr_credential_store_types.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd\_reset\_verification\_type\_example.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_reset\\_verification\\_type\\_example.png](http://wiki.servicenow.com/index.php?title=File:Pwd_reset_verification_type_example.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd\_verification\_sms\_processor.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_verification\\_sms\\_processor.png](http://wiki.servicenow.com/index.php?title=File:Pwd_verification_sms_processor.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd\_verification\_sms\_macro.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_verification\\_sms\\_macro.png](http://wiki.servicenow.com/index.php?title=File:Pwd_verification_sms_macro.png) *License:* unknown *Contributors:* George.rawlins

**Image:password\_reset\_verification\_type\_new.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Password\\_reset\\_verification\\_type\\_new.png](http://wiki.servicenow.com/index.php?title=File:Password_reset_verification_type_new.png) *License:* unknown *Contributors:* George.rawlins

**Image:pwd\_identification\_type\_new.png** *Source:* [http://wiki.servicenow.com/index.php?title=File:Pwd\\_identification\\_type\\_new.png](http://wiki.servicenow.com/index.php?title=File:Pwd_identification_type_new.png) *License:* unknown *Contributors:* George.rawlins

**Image:NewExtensionScript.png** *Source:* <http://wiki.servicenow.com/index.php?title=File:NewExtensionScript.png> *License:* unknown *Contributors:* George.rawlins