Uyuni 2026.11

Administration Guide



UYUNI

Chapter 1. Preface

Administration Guide

Uyuni 2026.11

This guide covers administration tasks like maintaining, monitoring, and customizing a Uyuni Server.

Publication Date: 2026-11-25

Table of Contents

1. Preface 1
2. Actions 9
   2.1. Recurring Actions 9
   2.2. Action Comments 11
   2.3. Action Chains 11
   2.4. Remote Commands 13
3. Ansible Integration 14
   3.1. Feature Overview 14
   3.2. Requirements and Basic Configuration 14
   3.3. Inventory Inspection 15
   3.4. Playbook Discovery 15
   3.5. Playbook Execution 15
   3.6. Setup Ansible Control Node 15
      3.6.1. Create Ansible Inventory Files 16
      3.6.2. Establish Communication with Ansible Nodes 17
   3.7. Compliance as Code 18
      3.7.1. Remediation Using an Ansible Playbook 19
4. Authentication Methods 21
   4.1. Authentication With Single Sign-On (SSO) 21
      4.1.1. Prerequisites 22
      4.1.2. Enable SSO 22
      4.1.3. Example SSO Implementation 23
   4.2. Authentication With PAM 28
      4.2.1. SSSD Configuration 28
5. Backup and Restore 31
   5.1. Back up Uyuni 31
      5.1.1. Full backup of Uyuni 31
      5.1.2. Partial backup of Uyuni 32
      5.1.3. Backing up extra volumes 33
      5.1.4. Perform a manual database backup 33
   5.2. Restore Uyuni from the existing backup 33
      5.2.1. Recommended steps after restoring a backup 34
   5.3. Database Backup Management 35
      5.3.1. Overview 35
      5.3.2. Prerequisites 35
      5.3.3. Enabling backups 35
      5.3.4. Checking backup status 36
      5.3.5. Maintaining backups (rebasing) 36
      5.3.6. Restoring from backup 36
      5.3.7. Disabling backups 37
      5.3.8. External documentation 37
6. Channel Management 38
   6.1. Channel Administration 38

6.2. Delete Channels 38
6.3. Custom Channels 39
6.3.1. Creating custom channels and repositories 39
6.3.2. Custom channel synchronization 44
6.3.3. Add packages and patches to custom channels 45
6.3.4. Manage custom channels 46
6.4. Third-party Channels 47
6.5. Configure Automatic Channels Synchronization 48
6.5.1. Verify channel synchronization status using the API 48
6.5.2. Verify automatic channels synchronization status 49
6.5.3. Set a channel to skip automatic synchronization 49
6.5.4. Manually synchronize a channel 50
7. Confidential Computing 51
7.1. Confidential Computing with Uyuni 51
7.2. Requirements 51
7.3. Limitations 52
7.4. Use Confidential Computing in Uyuni 52
7.4.1. Report Statuses 54
7.5. Related Topics 54
8. Content Lifecycle Management 55
8.1. Create a content lifecycle project 55
8.2. Filter types 56
8.2.1. Filter rule parameter 57
8.3. Filter templates 58
8.3.1. Live patching based on a SUSE product 58
8.3.2. Live patching based on a system 59
8.3.3. AppStream modules with defaults 60
8.4. Build a content lifecycle project 61
8.5. Promote environments 62
8.6. Assign clients to environments 62
8.7. Content Lifecycle Management Examples 63
8.7.1. Creating a Project for a Monthly Patch Cycle 63
8.7.2. Update an existing monthly patch cycle 66
8.7.3. Enhance a project with live patching 66
8.7.4. Switch to a new kernel version for live patching 68
8.7.5. AppStream filters 68
9. Content Staging 72
9.1. Enable Content Staging 72
9.2. Configure Content Staging 72
10. Disconnected Setup 74
10.1. Synchronize Channels and Repositories from SCC 74
10.1.1. Synchronize RMT 74
10.1.2. Synchronize SMT 75
10.2. Mandatory Channels 76
10.3. Disconnected Server 77
10.3.1. Deploy 77
10.3.2. Synchronize 77

11. Disk Space Management 79
11.1. Monitored Directories 79
11.2. Thresholds 80
11.2.1. Free space thresholds 80
11.3. Database Disk Check 82
11.4. Disable Space Checking 82
12. Image Building and Management 84
12.1. Image building overview 84
12.2. Container images 84
12.2.1. Requirements 84
12.2.2. Create a build host 84
12.2.3. Create an activation key for containers 85
12.2.4. Create an image store 87
12.2.5. Create an image profile 88
12.2.6. Build an image 91
12.2.7. Import an image 91
12.2.8. Troubleshooting 92
12.3. OS images 93
12.3.1. Requirements 93
12.3.2. Accessing Git repositories via an HTTP/HTTPS proxy when building images 94
12.3.3. Container-based Kiwi image build support 95
12.3.4. Create a build host 96
12.3.5. Create an activation key for OS images 98
12.3.6. Create an image store 99
12.3.7. Create an image profile 99
12.3.8. Build an image 102
12.3.9. Troubleshooting 103
12.3.10. Limitations 104
12.4. List of built images 104
13. Infrastructure maintenance tasks 105
13.1. Server 105
13.1.1. Client tools 105
13.2. Inter-Server Synchronization slave server 106
13.3. Monitoring server 106
13.4. Proxy 106
14. Live Patching with Uyuni 107
14.1. Set up Channels for Live Patching 107
14.1.1. Use spacewalk-manage-channel-lifecycle for Live Patching 107
14.2. Live Patching on SLES 15 109
14.3. Live Patching on SLES 12 110
15. Maintenance Windows 112
15.1. Maintenance Schedule Types 114
15.2. Restricted and Unrestricted Actions 115
16. Using mgr-sync 116
17. Monitoring with Prometheus and Grafana 118

17.1. Requirements 118
17.2. Prometheus and Grafana 118
17.2.1. Prometheus 118
17.2.2. Prometheus exporters 119
17.2.3. Grafana 119
17.3. Set up the monitoring server 119
17.3.1. Install Prometheus 119
17.3.2. Install Grafana 122
17.4. Configure Uyuni monitoring 124
17.4.1. Server self-monitoring 124
17.4.2. Monitoring managed systems 127
17.4.3. Change Grafana password 128
17.5. Network boundaries 129
17.5.1. Reverse proxy setup 129
17.6. Security 130
17.6.1. Generating TLS certificates 130
18. Organizations 133
18.1. Manage Organizations 133
18.1.1. Organization Users 134
18.1.2. Trusted Organizations 134
18.1.3. Configure Organizations 134
18.2. Manage States 134
18.2.1. Manage Configuration Channels 134
19. Patch Management 136
19.1. Retracted Patches 136
19.1.1. Channel Clones 136
19.1.2. Patch sharing 137
20. Using PTFs in Uyuni 138
20.1. Understanding PTF packages 138
20.2. Installing PTF packages 138
20.3. After PTF installation 140
20.4. Removing the patched version of a package 140
20.5. Removing the patched version of a package on the client 141
21. Generate Reports 142
21.1. Using spacewalk-report 142
21.2. spacewalk-report and the reporting database 142
21.3. List of available reports 143
21.4. Reporting Database 149
21.4.1. Setup 149
21.4.2. Database Schema 149
21.4.3. Data Generation and Update 150
21.4.4. Tuning 150
21.4.5. Visualizing Reporting Data with Grafana 151
22. Security 152
22.1. Auditing 152
22.1.1. CVE audits 152

22.1.2. OVAL 153
22.1.3. CVE Status 157
22.2. Set up a Client to Master Validation Fingerprint 157
22.3. Mirror Source Packages 158
22.4. System Security with OpenSCAP 159
22.4.1. About SCAP 159
22.4.2. Prepare clients for an SCAP scan 160
22.4.3. SCAP content management 161
22.4.4. Find OpenSCAP profiles 163
22.4.5. Tailoring file management 164
22.4.6. SCAP policies 165
22.4.7. Schedule SCAP scans 168
22.4.8. How scans work 171
22.4.9. Scan results 171
22.4.10. Removing SCAP Scan Results 172
22.4.11. Remediation 172
22.4.12. Navigation reference 175
22.4.13. Workflow example 176
22.5. Repository Metadata 176
23. Role-Based Access Control (RBAC) 180
23.1. Architectural scope 180
23.2. Key RBAC concepts 180
23.3. User roles in Uyuni 181
23.3.1. Predefined roles 181
23.3.2. Defining additional roles 181
23.4. Namespaces for fine-grained access 181
23.5. Managing RBAC 183
23.5.1. Managing RBAC via Web UI 183
23.5.2. Managing RBAC via API 183
23.6. RBAC best practices 184
24. SSL Certificates 185
24.1. Providing SSL certificates to the Uyuni containers 186
24.1.1. Podman 186
24.2. Self-Signed SSL Certificates 187
24.2.1. Re-Create Existing Server Certificates 187
24.2.2. Create a new CA and Server Certificates 188
24.3. Import SSL Certificates 188
24.3.1. Verify CA Basic Constraints 189
24.3.2. Import Certificates for New Installations 190
24.3.3. Import certificates for new proxy installations 190
24.3.4. Replace certificates 191
24.3.5. Replace certificates on a Proxy 192
24.4. Migrate SSL Certificate Authorities 193
24.4.1. Add the New CA to the Trust Bundle 194
24.4.2. Confirm the Clients Trust the New CA 195
24.4.3. Switch to the New CA 196
24.4.4. Update Proxies 197
24.5. HTTP Strict Transport Security 197

25. Subscription Matching 199
25.1. Pin Clients to Subscriptions 199
26. Task Schedules 201
26.1. Predefined task bunches 202
27. Tuning Changelogs 206
28. Users 207
28.1. Password requirements 207
28.2. Deactivate and delete accounts 208
28.3. User roles 208
28.4. Creating additional roles 209
28.5. User permissions and systems 211
28.6. Users and channel permissions 211
28.7. User default language 212
28.7.1. User default interface theme 213
29. Support 214
29.1. Create a service request number 214
29.2. Collect and upload support data from Uyuni to SUSE 214
30. Troubleshooting 216
30.1. Troubleshooting Autoinstallation 216
30.2. Troubleshooting Bootstrap Repository for End-of-Life Products 216
30.3. Troubleshooting Clients Cloned Salt Clients 217
30.4. Troubleshooting Container with Full Disk Event 217
30.5. Troubleshooting Corrupt Repositories 218
30.6. Troubleshooting Custom Channel with Conflicting Packages 218
30.7. Troubleshooting Disabling the FQDNS grain 219
30.8. Troubleshooting Disk Space 220
30.8.1. Disk Check Warnings for Missing Directories 220
30.9. Troubleshooting Firewalls 222
30.10. Troubleshooting high sync times between Uyuni Server and Proxy over WAN connections 223
30.11. Troubleshooting Inactive clients 225
30.12. Troubleshooting Inter-Server Synchronization 226
30.13. Troubleshooting libvirt events errors in minion log 226
30.14. Troubleshooting Local Issuer Certificates 228
30.15. Troubleshooting Login Timeouts 228
30.16. Troubleshooting Mail Configuration 229
30.17. Mass Machine_id Duplication 230
30.18. Troubleshooting Mounting /tmp with noexec 230
30.19. Troubleshooting Mounting /var/tmp with noexec 231
30.20. Troubleshooting Not Enough Disk Space 231
30.21. Troubleshooting Notifications 231
30.22. Troubleshooting Package Inconsistencies 231
30.23. Troubleshooting Passing Grains to a Start Event 232
30.24. Troubleshooting Proxy Connections and FQDN 232

30.25. Troubleshooting Registering Cloned Clients 233
30.26. Remote root login on SL Micro 235
30.27. Troubleshooting Registering Deleted Clients 236
30.28. Troubleshooting Registration from Web UI fails and does not show any errors 237
30.29. Troubleshooting Red Hat CDN Channel and Multiple Certificates 237
30.30. Troubleshooting Renaming Uyuni Server 238
30.30.1. Rename server 238
30.30.2. Reconfigure Proxy 240
30.31. Troubleshooting RPC Connection Timeouts 240
30.32. Troubleshooting Salt clients shown as down and DNS settings 241
30.33. Troubleshooting Salt SSH known hosts 242
30.34. Troubleshooting Schema Upgrade Fails 242
30.35. Troubleshooting Synchronization 243
30.36. Troubleshooting Taskomatic 245
30.37. Troubleshooting Failed Token Refresh 245
30.38. Troubleshooting Web UI Fails to Load 247
31. GNU Free Documentation License 248

Chapter 2. Actions

You can manage actions on your clients in a number of different ways:

- You can schedule automated recurring actions to apply the highstate or an arbitrary set of custom states to clients on a specified schedule.
- You can apply recurring actions to individual clients, to all clients in a system group, or to an entire organization.
- You can set actions to be performed in a particular order by creating action chains.
    - Action chains can be created and edited ahead of time, and scheduled to run at a time that suits you.
- You can also perform remote commands on one or more of your clients.
    - Remote commands allows you to issue commands to individual clients, or to all clients that match a search term.

2.1. Recurring Actions

You can apply recurring actions on individual clients, to a system group, or to all clients in an organization.

Currently, Uyuni supports the following action types as recurring actions:

- Highstate: Execute the highstate.
- Custom state: Execute a set of custom states. A custom state can be either an internal state provided by Uyuni, or a configuration channel created by a user.

For more information about configuration channels, see **Client Configuration Guide › Configuration Management**.

Procedure: Creating a new recurring action

1. To apply a recurring action to an individual client, navigate to Systems, click the client to configure schedules for, and navigate to the Recurring Actions tab.
2. To apply a recurring action to a system group, navigate to **Systems › System Groups**, select the group to configure schedules for, and navigate to Recurring Actions tab.
3. Click **[Create]**.

4. Select an action type from the Action Type dropdown.
5. Type a name for the new schedule.
6. Choose the frequency of the recurring action:
    - Hourly: Type the minute of each hour. For example, 15 runs the action at fifteen minutes past every hour.
    - Daily: Select the time of each day. For example, 01:00 runs the action at 0100 every day, in the timezone of the Uyuni Server.
    - Weekly: Select the day of the week and the time of the day, to execute the action every week at the specified time.
    - Monthly: Select the day of the month and the time of the day, to execute the action every month at the specified time.
    - Custom Quartz format: For more detailed options, enter a custom quartz string. For example, to run a recurring action at 0215 every Saturday of every month, enter:

```
0 15 2 ? * 7
```

7. OPTIONAL: Toggle the Test mode switch on to run the schedule in test mode.
8. For actions of type Custom state, select the states from the list of available states and click **[Save Changes]**. This will only save the current state selection and not the schedule.
9. In the next pane, drag and drop the selected states to put them in the execution order and click **[Confirm]**.
10. Click **[Create Schedule]** to save, and see the complete list of existing schedules.

Organization Administrators can set and edit recurring actions for all clients in the organization. Navigate to **Home › My Organization › Recurring Actions** to see all recurring actions that apply to the entire organization.

Uyuni Administrators can set and edit recurring actions for all clients in all organizations. Navigate to **Admin ›**

Organizations, select the organization to manage, and navigate to the Recurring Actions tab.

2.2. Action Comments

When you schedule an action from the Uyuni Web UI, you can add an optional comment to it. Type the comment in the Comment / Reason field of the scheduling options, for example a ticket number or the reason for the change. A comment can be up to 1000 characters long.

The comment is shown in the Comment / Reason column of the action lists under **Schedule**, in the event lists of a client under **Systems › Events**, and on the action details pages.

Scheduling API methods that support this feature accept an additional actionDescription argument. For details, see the API documentation.

2.3. Action Chains

If you need to perform a number of sequential actions on your clients, you can create an action chain to ensure the order is respected.

By default, most clients execute an action as soon as the command is issued. In some case, actions take a long time, which could mean that actions issued afterwards fail. For example, if you instruct a client to reboot, then issue a second command, the second action could fail because the reboot is still occurring. To ensure that actions occur in the correct order, use action chains.



For transactional update systems, an action chain is executed inside a single snapshot, until there is a reboot action. This can cause some limitations.

For more information, see **Client Configuration Guide › Clients Slemicro** and **Client Configuration Guide › Clients Opensuseleapmicro**.

You can use action chains on all clients. Action chains can include any number of these actions, in any order:

- **System Details › Remote Command**
- **System Details › Schedule System Reboot**
- **System Details › States › Highstate**
- **System Details › Software › Packages › List/Remove**
- **System Details › Software › Packages › Install**
- **System Details › Software › Packages › Upgrade**
- **System Details › Software › Patches**

- **System Details › Software › Software Channels**
- **System Details › Configuration**
- **Images › Build**



Action chains are user-specific. To see the action chain in the Web UI, you must sign in as the same user who created the action chain.

Procedure: Creating a new action chain

1. In the Uyuni Web UI, navigate to the first action you want to perform in the action chain. For example, navigate to System Details for a client, and click **[Schedule System Reboot]**.
2. Check the Add to field and select the action chain you want to add to:
    - If this is your first action chain, select new action chain.
    - If the action chain already exists, select it from the list.
    - If you already have existing action chains, but you want to create a new action chain, start typing a name for the new action chain to create it.
3. Confirm the action. The action is not performed immediately, it creates the new action chain, and a blue bar confirming this appears at the top of the screen.
4. Continue adding actions to your action chain by checking the Add to field and selecting the name of the action chain to add them to.
5. When you have finished adding actions, navigate to **Schedule › Action Chains** and selecting the action chain from the list.
6. Re-order actions by dragging them and dropping them into the correct position. Click the blue plus sign to see the clients an action is to be performed on. Click **[Save]** to save your changes.
7. Schedule a time for your action chain to run, and click **[Save and Schedule]**. If you leave the page without clicking either **[Save]** or **[Save and Schedule]** all unsaved changes are discarded.

If one action in an action chain fails, the action chain stops, and no further actions are executed.

You can see scheduled actions from action chains by navigating to **Schedule › Pending Actions**.

2.4. Remote Commands

Use this procedure to run a remote command via Salt.

Before you begin, ensure your client is subscribed to the appropriate tools child channel for its installed operating system. For more information about subscribing to software channels, see **Client Configuration Guide › Channels**.

- On transactional systems, remote commands can run on the live system or using transactional update. To select the execution mode from the system's Remote Command page, see **Reference Guide › Systems › Sd Remote Command**.
- Remote commands are run from the /tmp/ directory on the client. To ensure that remote commands work accurately, do not mount /tmp with the noexec option. For more information, see **Administration Guide › Troubleshooting › Troubleshooting Mounting /tmp with noexec**.
- All commands run from the Remote Commands page are executed as root on clients. Wildcards can be used to run commands across any number of systems. Always take extra care to check your commands before issuing them.

Procedure: Running remote commands on clients

1. Navigate to **Salt › Remote Commands**.
2. In the first field, before the @ symbol, type the command you want to issue.
3. In the second field, after the @ symbol, type the client you want to issue the command on. You can type the minion-id of an individual client, or you can use wildcards to target a range of clients.
4. Click **[Find targets]** to check which clients you have targeted, and confirm that you have used the correct details.
5. Click **[Run command]** to issue the command to the target clients.

Chapter 3. Ansible Integration

Ansible is a tool to manage computer client systems. For more information, see https://www.ansible.com.

Uyuni supports managing Ansible control nodes. For more information, see **Administration Guide › Ansible Setup Control Node › At.ansible.overview**.

The supported version for the Ansible Control Node is Ansible 11.3. It should be obtained from the operating system vendor's official repositories. For example, on SUSE Linux Enterprise 15 SP6 and SP7, Ansible is available through the Systems Management Module. For Control Nodes running operating systems other than SUSE Linux Enterprise, use Ansible shipped together with that distribution.

Ansible software is also available for Uyuni Proxy and Uyuni for Retail Branch Server.

3.1. Feature Overview

Uyuni enables system administrators to operate their Ansible Control Nodes. Supported features are:

- inspection of inventory files
- discovery of playbooks
- execution of playbooks

For more information:

- The inventory is a sorted list of managed Ansible nodes. For more information about organizing an inventory, see https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html.
- Playbooks are a way to describe how the inventory is to be managed. For more information about playbooks, see https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html.

3.2. Requirements and Basic Configuration

To use Ansible features, you need to register the already existing Ansible Control Node to the Uyuni Server. In the Web UI, on the **System Details › Properties** page of the registered system, you must enable the Ansible Control Node system type of the Add-on System Types list.

Enabling the Ansible Control Node system type ensures that the ansible package is installed on the system by adding it in the highstate and activates the Ansible features under the **System Details › Ansible** tab.

As the next step, configure the paths to your Ansible playbook directories and inventory files on the **System Details › Ansible › Control Node** page. As an inventory path, you can use the standard Ansible inventory path /etc/ansible/hosts. As a playbook directory, you can use any directory on the control node, where your

playbook files are stored. A playbook directory either contains .yml files or subdirectories with .yml files.

For installing and setting up an Ansible Control Node, see **Administration Guide › Ansible Setup Control Node**.

3.3. Inventory Inspection

After defining an inventory path, you can use Uyuni to inspect its contents.

Procedure: Inspecting an Inventory from the Web UI

1. In the Uyuni Web UI, navigate to **System Details › Ansible › Inventories**
2. Click an inventory path to execute the inventory inspection on the Control Node in real-time.

3.4. Playbook Discovery

After defining a playbook directory, you can discover playbooks on the **System Details › Ansible › Playbooks** page.

As with the inventory inspection, the playbook discovery operations run on the Control Node in real-time.

3.5. Playbook Execution

You can schedule a playbook execution from the **System Details › Ansible › Playbooks** page. After selecting the playbook you wish to execute, you can select the inventory file for the execution from the Inventory Path drop-down menu of the Schedule Playbook Execution dialog. If you do not select any item, the default inventory configured in your Control Node will be used. The drop-down menu is populated with the inventories you defined in your Inventory paths and with inventories that have been locally discovered in your playbook directories. These are displayed as Custom Inventory items in the playbook details. You can also enter an arbitrary inventory path.

You can also pass variables to the playbook execution to override previously defined variables. To pass variables, locate the Playbook Content section and click **[Edit variables]**. In the Edit Variables dialog, you can add or edit variables as a list, dictionary, string, or boolean. Alternatively, click **[Edit YAML]** to edit the variables directly as YAML. Click **[Save]** to apply the variables to the playbook content.

Afterwards, you choose the time of the playbook execution or select an action chain. Eventually, Uyuni executes the playbook as an action on the Control Node. You can see the result of the operation on the action details page.

3.6. Setup Ansible Control Node

To set up an Ansible control node, execute the following steps from the Uyuni Web UI.



To configure a client as the Ansible Control Node, the Ansible package must be installed on that system. Usually, the Ansible package should be obtained from the operating system vendor's official repositories. For example, on SUSE Linux Enterprise 15 SP6 and SP7, Ansible is available through the Systems Management Module.

Procedure: Setting up Ansible Control Node on a SUSE Linux Enterprise 15 SP6 or SP7 system

1. In the Uyuni Web UI, navigate to **Admin › Setup Wizard › Products**, verify that SUSE Linux Enterprise Server 15 SP6 x86_64 (or later) with the Systems Management Module and the required Python 3 Module are selected and synchronized.
2. Deploy a SUSE Linux Enterprise 15 SP6 (or later) client.
3. In the Uyuni Web UI, navigate to the **Systems › Overview** page of the client. Select **Software › Software Channels** and subscribe the client to the SUSE Linux Enterprise Server 15 SP6 x86_64 (or later SP), Systems Management Module and Python 3 Module channels.
4. Select **Details › Properties** of your client. From the Add-On System Types list enable Ansible Control Node and click **[Update Properties]**.
5. Navigate to the client overview page, select **State › Highstate**, and click **[Apply Highstate]**.
6. Select the **Events** tab and verify the status of the highstate.



If you want to install a newer Ansible on a SUSE Linux Enterprise 15 SP4 or SP5 client, you must enable the Python 3 Module.



Newer versions of Ansible no longer support managing nodes with outdated Python versions. If a managed node still defaults to an older Python version, you may encounter connection errors or failures during playbook runs. To address this, user should upgrade Python on the managed node, if possible and set the correct Python interpreter in the Ansible inventory or configuration.

3.6.1. Create Ansible Inventory Files

Ansible Integration tools deploy a playbook as an inventory file. Create one inventory file for each operating

system listed in Table 1.

Procedure: Creating Ansible Inventory Files

1. Create and add your hosts to an inventory file to be managed by Ansible. The default path for an Ansible inventory is /etc/ansible/hosts.

   Listing 1. Inventory Example

   ```
   client240.mgr.example.org
   client241.mgr.example.org
   client242.mgr.example.org
   client243.mgr.example.org
   ansible_ssh_private_key_file=/etc/ansible/some_ssh_key

   [mygroup1]
   client241.mgr.example.org
   client242.mgr.example.org

   [mygroup2]
   client243.mgr.example.org

   [all:vars]
   ansible_ssh_private_key_file=/etc/ansible/my_ansible_private_key
   ```

2. In the Uyuni Web UI, from the Ansible tab navigate to **Ansible › Control Node** to add inventory files to the control node.

3. Under the Playbook Directories section add /usr/share/scap-security-guide/ansible to the Add a Playbook Directories field and click **[Save]**.

4. Under Inventory Files add your inventory file locations to the Add an Inventory file field and click **[Save]**.

   Listing 2. Examples

   ```
   /etc/ansible/sles15
   /etc/ansible/sles12
   /etc/ansible/centos7
   ```

   For additional playbook examples, see https://github.com/ansible/ansible-examples.

3.6.2. Establish Communication with Ansible Nodes

Procedure: Establishing communication with Ansible Nodes

1. Create the SSH keys that you are using in your inventory.

```
ssh-keygen -f /etc/ansible/my_ansible_private_key
```

2. Copy the generated SSH keys to the Ansible managed clients. Example:

```
ssh-copy-id -i /etc/ansible/my_ansible_public_key
root@client240.mgr.example.org
```

3. Declare the private key in /etc/ansible/ansible.cfg as follows:

```
private_key_file = /etc/ansible/my_ansible_private_key
```

   Replace my_ansible_private_key with the name of the file containing the private key.

4. Test that Ansible is working by executing the following commands from the control node:

```
ansible all -m ping
ansible mygroup1 -m ping
ansible client240.mgr.example.org -m ping
```

You may now run remediations. For more information, see **Administration Guide › Ansible Compliance As Code**.

3.7. Compliance as Code

This document provides insight on running compliance as code remediation using an Ansible Playbook.

For more infomation about running compliance as code remediation using a bash script, see **Administration Guide › Remediation**.

For executing remediations you need to install the SCAP security guide package on the Ansible control node.

Procedure: Installing the SCAP security guide package

1. From **Systems › Overview**, select the client. Then click **Software › Packages › Install**.

2. Search for scap-security-guide and install the package suitable for your system. See the following table for package distribution requirements:

Table 1. SCAP security guide package requirements

Package name	Supported Systems
scap-security-guide	openSUSE, SLES12, SLES15
scap-security-guide-redhat	CentOS 7, CentOS 8, Fedora, Oracle Linux 7, Oracle Linux 8, RHEL7, RHEL8, RHEL9, Red Hat OpenStack Platform 10, Red Hat OpenStack Platform 13, Red Hat Virtualization 4, Scientific Linux
scap-security-guide-debian	Debian 12
scap-security-guide-ubuntu	Ubuntu 22.04



For Debian 13, Ubuntu 24.04, and Ubuntu 26.04, SUSE does not provide custom security guide packages. You must use upstream packages (ssg-debian and ssg-debderived respectively) from their respective community repositories. Please note that these upstream guides are not officially supported by SUSE.

3.7.1. Remediation Using an Ansible Playbook

An Ansible control node is required. For more information, see **Administration Guide › Ansible Setup Control Node**.

The following procedure will guide you through running remediation using an Ansible playbook.

Procedure: Run Remediation Using an Ansible Playbook

1. From the control node system menu select **Ansible › Playbooks**, and click a playbook, for example:

```
sle15-playbook-stig.yml
```

2. To run the playbook, select the Inventory Path for the clients, for example:

```
/etc/ansible/sles15
```

Click **[Schedule]**.

3. Check the status of the scheduled event under the Events tab.

In case playbooks are in a different directory, you can follow the link to Setup Ansible Control Node to find out how to add it.

Chapter 4. Authentication Methods

Uyuni supports several different authentication methods. This section discusses pluggable authentication modules (PAM) and single sign-on (SSO).

4.1. Authentication With Single Sign-On (SSO)

Uyuni supports single sign-on (SSO) by implementing the Security Assertion Markup Language (SAML) 2 protocol.

Single sign-on is an authentication process that allows a user to access multiple applications with one set of credentials. SAML is an XML-based standard for exchanging authentication and authorization data. A SAML identity service provider (IdP) provides authentication and authorization services to service providers (SP), such as Uyuni. Uyuni exposes three endpoints which must be enabled for single sign-on.

SSO in Uyuni supports:

- Log in with SSO.
- Log out with service provider-initiated single logout (SLO), and Identity service provider single logout service (SLS).
- Assertion and nameId encryption.
- Assertion signatures.
- Message signatures with AuthNRequest, LogoutRequest, and LogoutResponses.
- Enable an Assertion consumer service endpoint.
- Enable a single logout service endpoint.
- Publish the SP metadata (which can be signed).

SSO in Uyuni does not support:

- Product choosing and implementation for the identity service provider (IdP).
- SAML support for other products (check with the respective product documentation).

For an example implementation of SSO, see **Administration Guide › Auth Methods Sso Example**.



If you change from the default authentication method to single sign-on, the new SSO credentials apply only to the Web UI. Client tools such as mgr-sync or spacecmd continue to work with the default authentication method only.

4.1.1. Prerequisites

Before you begin, you need to have configured an external identity service provider with these parameters. Check your IdP documentation for instructions.



The mapping between the IdP user and the Uyuni user is specified in a SAML:Attribute. The SAML:Attribute must be configured in the IdP and must be passed to Uyuni in the SAML authentication. The attribute must be named uid and must contain the Uyuni user mapped to it after login. The Uyuni user must be created before you activate single sign-on.

You need these endpoints:

- Assertion consumer service (or ACS): an endpoint to accept SAML messages to establish a session into the Service Provider. The endpoint for ACS in Uyuni is: https://server.example.com/rhn/manager/sso/acs
- Single logout service (or SLS): an endpoint to initiate a logout request from the IdP. The endpoint for SLS in Uyuni is: https://server.example.com/rhn/manager/sso/sls
- Metadata: an endpoint to retrieve Uyuni metadata for SAML. The endpoint for metadata in Uyuni is: https://server.example.com/rhn/manager/sso/metadata

After the authentication with the IdP using the user orgadmin is successful, you are logged in to Uyuni as the orgadmin user, provided that the orgadmin user exists in Uyuni.

4.1.2. Enable SSO



Using SSO is mutually exclusive with other types of authentication: it is either enabled or disabled. SSO is disabled by default.



Use mgrctl term before running steps inside the server container.

Procedure: Enabling SSO

1. If your users do not yet exist in Uyuni, create them first.
2. Edit /etc/rhn/rhn.conf and add this line at the end of the file:

```
java.sso = true
```

3. Find the parameters you want to customize in /usr/share/rhn/config-defaults/rhn_java_sso.conf. Insert the parameters you want to customize into /etc/rhn/rhn.conf and prefix them with java.sso. For example, in /usr/share/rhn/config-defaults/rhn_java_sso.conf find:

```
onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-HOSTNAME-OR-IP/rhn/manager/sso/acs
```

To customize it, create the corresponding option in /etc/rhn/rhn.conf by prefixing the option name with java.sso.:

```
java.sso.onelogin.saml2.sp.assertion_consumer_service.url = https://YOUR-PRODUCT-HOSTNAME-OR-IP/rhn/manager/sso/acs
```

To find all the occurrences you need to change, search in the file for the placeholders YOUR-PRODUCT and YOUR-IDP-ENTITY. Every parameter comes with a brief explanation of what it is meant for.

4. Restart the spacewalk service to pick up the changes:

```
mgradm restart
```

When you visit the Uyuni URL, you are redirected to the IdP for SSO where you are requested to authenticate. Upon successful authentication, you are redirected to the Uyuni Web UI, logged in as the authenticated user. If you encounter problems with logging in using SSO, check the Uyuni logs for more information.

4.1.3. Example SSO Implementation

In this example, SSO is implemented by exposing three endpoints with Uyuni, and using Keycloak 26.5.3 or later as the identity service provider (IdP).

To set up Single Sign-On (SSO) with Uyuni and Keycloak, you first configure the Keycloak identity provider, obtain its public SAML certificate, and set up the client and users. After that, you configure the Uyuni Server to trust Keycloak.



This example is provided for illustrative purposes only. SUSE does not recommend or support third-party identity service providers, and is not affiliated with Keycloak. For Keycloak support, see https://www.keycloak.org/.

You can install Keycloak directly on your machine, or run it in a container. In this example, we run Keycloak in a Podman container. For more information about installing Keycloak, see the Keycloak documentation at https://www.keycloak.org/guides#getting-started.

4.1.3.1. Setting up the identity service provider (Keycloak)

First, run Keycloak, create a realm, obtain the SAML public certificate, and configure the client.

Procedure: Setting up Keycloak

1. Install Keycloak in a Podman container, according to the Keycloak documentation.
2. Run the container using the -td argument to ensure the process remains running:

```
podman run -td --name keycloak -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e
KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:26.1.1 start-dev
```

3. Sign in to the Keycloak Admin Console as the admin user.
4. Create an authentication realm:
   - In the top-left corner, click the realm selector dropdown (showing master) and select Manage realms.
   - On the Manage realms page, click the **[Create realm]** button in the top-right corner.
   - In the Realm name field, enter a name (for example, MLM).
   - Click **[Create]**.
5. Obtain the SAML public certificate:
   - From the left navigation menu, click **Realm settings**.
   - Click the Keys tab.
   - Find the active **RSA** key (with provider rsa-generated) in the active keys table.
   - Click the **[Certificate]** button in that row.
   - Copy the entire Base64-encoded certificate from the modal that opens. Keep this certificate handy for the server configuration.
6. Add Uyuni as a Client:
   - From the left navigation menu, click **Clients**.
   - Click **[Create client]**.

- In the Client type field, select SAML.
- In the Client ID field, enter the Uyuni Metadata endpoint. For example: https://<FQDN_MLM>/rhn/manager/sso/metadata
- Click **[Next]**.
- In the Valid redirect URIs field, enter the Uyuni Assertion Consumer Service (ACS) endpoint. For example: https://<FQDN_MLM>/rhn/manager/sso/acs
- Click **[Save]**.

7. Fine-tune client settings:
   - Under the Settings tab, scroll down to the SAML capabilities section:
     - Toggle Sign assertions to On.
     - In the Signature algorithm field, select RSA_SHA1.
     - In the SAML Signature Key Name field, select Key ID.
     - Click **Save**.
   - Under the Keys tab:
     - Toggle Client signature required to Off.
   - Under the Advanced tab, scroll down to the Fine Grain SAML Endpoint Configuration section:
     - In both the Assertion Consumer Service POST Binding URL and Assertion Consumer Service Redirect Binding URL fields, enter the Uyuni ACS endpoint. For example: https://<FQDN_MLM>/rhn/manager/sso/acs
     - In the Logout Service POST Binding URL field, enter the Uyuni SLS endpoint. For example: https://<FQDN_MLM>/rhn/manager/sso/sls
     - In the Logout Service Redirect Binding URL field, enter the root URL of your Uyuni instance. For example: https://<FQDN_MLM>/

- Click **Save**.

8. Configure Client Scope and Mappers:
   - From the left navigation menu, click **Client scopes**.
   - Click role_list in the list.
   - Click the Mappers tab, and then click the role list mapper.
   - Toggle Single Role Attribute to On, and click **[Save]**.
   - Go back to **Clients**, click your Uyuni client ID (https://<FQDN_MLM>/rhn/manager/sso/metadata), and click the Client scopes tab.
   - Click the dedicated scope (for example, https://<FQDN_MLM>/rhn/manager/sso/metadata-dedicated).
   - Click **[Configure a new mapper]**.
   - Select User Attribute from the list, and configure the mapper with these details:
     - In the Name field, enter uid.
     - In the Property field, select username.
     - In the SAML Attribute Name field, enter uid.
   - Click **[Save]**.
9. Create Users:
   - From the left navigation menu, click **Users**.
   - Click **[Add user]**.
   - In the Username field, enter a username that matches an existing Uyuni user (for example, the administrative user).
   - Click **[Create]**.
   - Click the Credentials tab, click **[Set password]**, set a password for the user, and toggle Temporary to Off, click **[Save]** and confirm by clicking **[Save password]**.

4.1.3.2. Setting up the Uyuni server

After Keycloak is ready, configure the Uyuni Server to use single sign-on.



Ensure that the users whom you intend to authenticate using SSO already exist in Uyuni. You can create users manually in the Uyuni Web UI, or use the command-line interface or API (using PAM-enabled or passwordless authentication).

Procedure: Setting up the Uyuni server

1. On the Uyuni Server, open the /etc/rhn/rhn.conf configuration file.
2. Add the following parameters. Replace <FQDN_MLM> with the fully qualified domain name of your Uyuni installation, <FQDN_IDP> with the FQDN of your Keycloak server, <REALM> with your realm name (for example, MLM), and <CERTIFICATE> with the SAML public certificate Base64-encoded string obtained from Keycloak in the previous steps:

```
java.sso = true

java.sso.onelogin.saml2.debug = true

java.sso.onelogin.saml2.sp.entityid                       =
https://<FQDN_MLM>/rhn/manager/sso/metadata
java.sso.onelogin.saml2.sp.assertion_consumer_service.url =
https://<FQDN_MLM>/rhn/manager/sso/acs
java.sso.onelogin.saml2.sp.single_logout_service.url      =
https://<FQDN_MLM>/rhn/manager/sso/sls

java.sso.onelogin.saml2.idp.entityid                   =
http://<FQDN_IDP>:8080/realms/<REALM>
java.sso.onelogin.saml2.idp.single_sign_on_service.url =
http://<FQDN_IDP>:8080/realms/<REALM>/protocol/saml
java.sso.onelogin.saml2.idp.single_logout_service.url  =
http://<FQDN_IDP>:8080/realms/<REALM>/protocol/saml

java.sso.onelogin.saml2.idp.x509cert = -----BEGIN CERTIFICATE-----
<CERTIFICATE> -----END CERTIFICATE-----

# Organization Metadata (Optional)
java.sso.onelogin.saml2.organization.name = SUSE Multi-Linux Manager
java.sso.onelogin.saml2.organization.displayname = SUSE Multi-Linux Manager
java.sso.onelogin.saml2.organization.url = https://<FQDN_MLM>
java.sso.onelogin.saml2.organization.lang =

# Contacts (Optional)
java.sso.onelogin.saml2.contacts.technical.given_name = SUSE Multi-Linux
Manager Admin
java.sso.onelogin.saml2.contacts.technical.email_address = admin@example.com
java.sso.onelogin.saml2.contacts.support.given_name = SUSE Multi-Linux Manager
Admin
java.sso.onelogin.saml2.contacts.support.email_address = admin@example.com
```

3. Save your changes and restart the spacewalk service to apply them:

```
mgradm restart
```

4. Navigate to the Uyuni Web UI. You should be redirected to the Keycloak login screen, and successfully log in back to Uyuni upon authentication.

4.1.3.3. Troubleshooting

If you encounter issues when authenticating with single sign-on, use these steps to diagnose:

- **Verify Logs:** Look for SSO errors in /var/log/rhn/rhn_web_ui.log. Detailed logging is enabled when java.sso.onelogin.saml2.debug = true is set in /etc/rhn/rhn.conf.
- **User Not Found:** If you see "User not found" or redirection fails after Keycloak authentication, ensure the username in Keycloak matches the username in Uyuni exactly. Users must be pre-created in Uyuni.
- **Signature and certificate errors:** If you see "Signature verification failed", verify that the certificate in java.sso.onelogin.saml2.idp.x509cert matches the active RSA certificate displayed under the Keys tab in Keycloak Realm Settings, and ensure that Sign assertions is toggled to On in the client settings.
- **Invalid Redirect URI:** If Keycloak displays "Invalid parameter: redirect_uri" on redirection, verify that Valid redirect URIs in client settings in Keycloak is set to https://<FQDN_MLM>/rhn/manager/sso/acs.

4.2. Authentication With PAM

Uyuni supports network-based authentication systems using pluggable authentication modules (PAM) using SSSD. PAM is a suite of libraries that allows you to integrate Uyuni with a centralized authentication mechanism, eliminating the need to remember multiple passwords. Uyuni supports LDAP, Kerberos, and other network-based authentication protocols.

4.2.1. SSSD Configuration

Procedure: Configuring SSSD

1. In the Uyuni Web UI, navigate to **Users › Create User** and enable a new or existing user to authenticate with PAM.



In usernames, additionally to alphanumeric characters, -, _, ., and @ are allowed.

2. Check the Pluggable Authentication Modules (PAM) checkbox.

3. Configure SSSD in the server container. At the command prompt of the Uyuni container host, as root, enter the server container:

```
mgrctl term
```

4. Inside the container, execute the following steps:

   a. Edit /etc/sssd/sssd.conf according to your configuration. For an example, see **Administration Guide › Auth Methods Pam › Auth Methods Pam Ad**.

   b. When done, exit the container:

```
exit
```

5. Restart Uyuni using:

```
mgradm restart
```



Changing the password in the Uyuni Web UI changes only the local password on the Uyuni Server. If PAM is enabled for that user, the local password might not be used at all. In the above example, for instance, the Kerberos password is not changed. Use the password change mechanism of your network service to change the password for these users.

For more information about PAM configuration, see the SUSE Linux Enterprise Server Security Guide. The Security Guide contains a generic example that also works for other network-based authentication methods. It also describes how to configure an Active Directory (AD) service. For more information, see https://documentation.suse.com/sles/15-SP7/html/SLES-all/part-auth.html.

4.2.1.1. LDAP Integration with Active Directory Example

For LDAP integration with Active Directory, you can use the following example.

In the code snippet, change the following placeholders according to your environment:

$domain

Your domain name

$ad_server

FQDN of the AD server if it is not auto-detected from the $domain $uyuni-hostname: The name of the machine this AD client is supposed to be known. If not set, it will be uyuni-server.mgr.internal.

Example snippet for /etc/sssd/sssd.conf:

```
[sssd]
config_file_version = 2
services = nss, pam
domains = $domain

[nss]

[pam]

[domain/$domain]
id_provider = ad
chpass_provider = ad
access_provider = ad
auth_provider = ad

ad_domain = $domain
ad_server = $ad_server
ad_hostname = $uyuni-hostname

ad_gpo_map_network = +susemanager

krb5_keytab = FILE:/etc/rhn/krb5.conf.d/krb5.keytab
krb5_ccname_template = FILE:/tmp/krb5cc_%{uid}
```

Chapter 5. Backup and Restore

This chapter contains information on the files you need to back up. With the built-in backup and restore solution (mgradm backup) you create Uyuni backups. Information about restoring from your backups in the case of a system failure completes this chapter.

Because Uyuni relies on a database as well as the installed program and configurations, it is important to back up all components of your installation. Back up your Uyuni installation regularly to prevent data loss and enable quick recovery.



Regardless of the backup method you use, you must have available at least three times the amount of space your current installation uses. Running out of space can result in backups failing, so check this often.

5.1. Back up Uyuni

The most comprehensive method for backing up your Uyuni installation is to use mgradm backup create command. This can save you time in administering your backup, and can be faster to reinstall and re-synchronize in the case of failure. However, this method requires significant disk space and could take a long time to perform the backup.

mgradm backup create command performs backup to a directory. This directory can be both local or mounted remote storage.

mgradm backup create command allows various customizations of the content of the backup. For all available options, see mgradm backup create --help.

5.1.1. Full backup of Uyuni

A full backup of the Uyuni consists of backing up the following components:

- Uyuni volumes
- database volumes
- podman network configuration
- podman secrets
- Uyuni systemd services
- Uyuni container images



The Uyuni service is automatically stopped for the time it takes to create a full backup. The downtime can be significant. After backup is done, service is automatically restarted.

Procedure: Creating full backup with mgradm backup create

1. On the container host, as root, create backup with:

```
mgradm backup create $path
```

   Replace $path by the path to the backup location.

5.1.2. Partial backup of Uyuni

mgradm backup create tool allows creating partial backups. It is possible to skip individual or all volumes, skip database backup and images.

Particularly when database backup is skipped, backup is created without stopping Uyuni services and can act as a one phase in two phase backup procedure.



Partial backups are only considering a part of the data, and do not take potential dependencies with other parts which may not have been backed up in consideration. Therefore they cannot guarantee backup/restore consistency.

Procedure: Creating partial backup by skipping database backup

1. On the container host, as root, create backup with:

```
mgradm backup create --skipdatabase $path
```

   Replace $path by the path to the backup location.

Procedure: Creating partial backup by skipping a volume

1. On the container host, as root, create backup with:

```
mgradm backup create --skipvolumes $volumes $path
```

   Replace $path by the path to the backup location.

   Replace $volumes by the name of the volume name to be excluded from the backup, or by a comma separated list of volumes to be excluded.

   Use all to skip all volumes, except database volumes.

5.1.3. Backing up extra volumes

mgradm backup command uses internal list of Uyuni volumes. If additional volumes were configured during the installation, or additional volumes should be added to the backup, they need to be specified using --extravolumes $volumes.

Procedure: Creating backup with additional custom volume

1. On the container host, as root, create backup with:

```
mgradm backup create --extravolumes $volume $path
```

   Replace $path by the path to the backup location.

   Replace $volumes by the name of the volume name to be included in the backup. or by a comma separated list of volumes to be included.

5.1.4. Perform a manual database backup

Procedure: Performing a manual database backup

1. Allocate permanent storage space for your backup.
2. At the command prompt of the Uyuni container host, as root, use:

```
mgradm backup create --skipvolumes all --skipconfig --skipimages $path
```

5.2. Restore Uyuni from the existing backup

Restoring Uyuni from the existing backup will enumerate backup for volumes, images and configuration to restore. Unlike in backup create scenario, restore operation is not using an internal volume list, but automatically detect every volume or image present in the backup.

After the list of items to restore is gathered, presence and integrity check is performed. Presence check ensures backup restore will not accidentally overwrite existing volumes, image or configurations. Integrity check is done by computing backup items checksums.

After both checks are successful, actual backup restore is performed.



Uyuni services are not automatically started after backup restore is finished.

Procedure: Restoring from an existing backup

1. On the container host, as root, re-deploy the Uyuni Server with:

```
mgradm stop
mgradm backup restore $path
mgradm start
```

   Replace $path by the path to the backup location.

Verification of the backup can be a time-consuming operation. If backup integrity is ensured by other means, verification can be skipped by using --skipverify option.

If for some reason it is needed to skip restoring a volume present in the backup, --skipvolumes $volumes option can be used.

5.2.1. Recommended steps after restoring a backup

Procedure: Recommended steps after Uyuni restore

1. Re-synchronize your Uyuni repositories using either the Uyuni Web UI, or with the mgr-sync tool at the command prompt in the container. You can choose to re-register your product, or skip the registration and SSL certificate generation sections.
2. On the container host, check whether you need to restore /var/lib/containers/storage/volumes/var-spacewalk/_data/packages/. If /var/lib/containers/storage/volumes/var-spacewalk/_data/packages/ was not in your backup, you need to restore it. If the source repository is available, you can restore `/var/lib/containers/storage/volumes/var-spacewalk/_data/packages/` u with a complete channel synchronization:

```
mgrctl exec -ti -- mgr-sync refresh --refresh-channels
```

3. Schedule the re-creation of search indexes next time the rhn-search service is started.

   This command produces only debug messages, it does not produce error messages.

On the container host, enter:

```
mgrctl exec -ti -- rhn-search cleanindex
```

5.3. Database Backup Management

This guide describes how to manage online database backups for the Uyuni server using mgradm. The backup system is based on PostgreSQL continuous archiving Write-Ahead Logging (WAL).



- To ensure the system can be fully recovered, perform regular backups of both the database and other system volumes.
- Regularly copy your backup volumes and WAL files to an external storage device or an off-site location for disaster recovery.

5.3.1. Overview

Continuous archiving reduces the risk of data loss by constantly backing up the transaction logs (WAL) to a dedicated volume.

The backup system utilizes a dedicated Podman volume named var-pgsql-walbackup to store:

- A base backup (a full snapshot of the database).
- WAL segments (incremental changes since the last base backup).



Point-in-time recovery is currently not supported by mgradm and requires a manual workflow.

Only full recovery is supported by mgradm tool. The restore procedure will restore everything from the backup location.

5.3.2. Prerequisites

- mgradm tool installed on the host.
- Uyuni server running in a Podman environment.

5.3.3. Enabling backups

To enable continuous archiving, run the following command:

```
mgradm backup db enable
```

This command:

- Verifies that the backup is not already enabled.
- Configures PostgreSQL (postgresql.conf) to enable archive_mode and sets the archive_command.
- Configures the var-pgsql-walbackup volume for the database service.
- Restarts the database service to apply changes.
- Performs an initial base backup inside the container.

If the backup is already configured but you need to re-initialize it, use the --force flag.

5.3.4. Checking backup status

You can verify the current state of the database backup system at any time:

```
mgradm backup db status
```

The command will report one of the following statuses:

- **enabled**: Backup is correctly configured and active.
- **disabled**: Continuous archiving is explicitly turned off.
- **misconfigured**: There is a discrepancy between the configuration and the runtime state (for example, missing volume, incorrect archive command).

5.3.5. Maintaining backups (rebasing)

Over time, the number of WAL segments can grow significantly, consuming disk space and increasing restoration time. It is recommended to periodically "rebase" the backup by creating a new base backup and starting a new WAL chain.

```
mgradm backup db rebase
```

This command creates a fresh full snapshot of the database without requiring a service restart.



Regularly rebase your database backups using command mgradm backup db rebase to minimize the time required for restore operations.

5.3.6. Restoring from backup

Restoring from a backup is a **destructive operation** that replaces the current database data with the content of the backup.

```
mgradm backup db restore
```

The restoration process:

- Stops the database service.
- Clears the current database data directory.
- Extracts the base backup from the backup volume.
- Configures PostgreSQL for recovery.
- Starts the database service.
- Replays the WAL segments to bring the database to the latest possible state.

The command will wait until the database has fully recovered and is ready to accept connections.



Ensure you have a recent base backup and all necessary WAL segments in the var-pgsql-walbackup volume before performing a restore.

5.3.7. Disabling backups

To turn off continuous archiving:

```
mgradm backup db disable
```

By default, this command disables archiving in the configuration and restarts the database but preserves the backup volume.

To also remove the backup volume and all stored backups, use:

```
mgradm backup db disable --purge-volume
```



Regularly test your restore procedures to verify data integrity and ensure you are prepared for a recovery.

5.3.8. External documentation

For more information on PostgreSQL, see PostgreSQL 16: Continuous Archiving and Point-in-Time Recovery (PITR).

Chapter 6. Channel Management

Channels are a method of grouping software packages.

In Uyuni, channels are grouped into base and child channels, with base channels grouped by operating system type, version, and architecture, and child channels being compatible with their related base channel. When a client has been assigned to a base channel, it is only possible for that system to install the related child channels. Organizing channels in this way ensures that only compatible packages are installed on each system.

Software channels use repositories to provide packages. The channels mirror the repositories in Uyuni, and the package names and other data are stored in the Uyuni database. You can have any number of repositories associated with a channel. The software from those repositories can then be installed on clients by subscribing the client to the appropriate channel.

Clients can only be assigned to one base channel. The client can then install or update packages from the repositories associated with that base channel and any of its child channels.

Uyuni provides a number of vendor channels, which provide you everything you need to run Uyuni. Uyuni Administrators and Channel Administrators have channel management authority, which gives them the ability to create and manage their own custom channels. If you want to use your own packages in your environment, you can create custom channels. Custom channels can be used as a base channel, or you can associate them with a vendor base channel.

For more on creating custom channels, see **Administration Guide › Custom Channels**.

6.1. Channel Administration

By default, any user can subscribe channels to a system. You can implement restrictions on the channel using the Web UI.

Procedure: Restricting Subscriber Access to a Channel

1. In the Uyuni Web UI, navigate to **Software › Channel List**, and select the channel to edit.
2. Locate the Per-User Subscription Restrictions section and check Only selected users within your organization may subscribe to this channel. Click **[Update]** to save the changes.
3. Navigate to the Subscribers tab and select or deselect users as required.

6.2. Delete Channels

You can delete vendor software channels from the command prompt.

Procedure: Deleting Vendor Channels

1. On the Uyuni Server, at the command prompt, as root, list the available vendor channels, and make a note of the channel you want to delete:

```
mgr-sync list channels
```

2. Delete the channel:

```
spacewalk-remove-channel -c <channel-name>
```

   - For information about deleting custom channels, see **Administration Guide › Custom Channels**.

6.3. Custom Channels

Custom channels give you the ability to create your own software packages and repositories, which you can use to update your clients. They also allow you to use software provided by third party vendors in your environment.

This section gives more detail on how to create, administer, and delete custom channels. You must have administrator privileges to be able to create and manage custom channels.

6.3.1. Creating custom channels and repositories

Before you create a custom channel, determine which base channel you want to associate it with, and which repositories you want to use for content.

If you have custom software packages that you need to install on your client systems, you can create a custom child channel to manage them. You need to create the channel in the Uyuni Web UI and create a repository for the packages, before assigning the channel to your systems.



Do not create child channels containing packages that are not compatible with the client system.

You can select a vendor channel as the base channel if you want to use packages provided by a vendor. Alternatively, select none to make your custom channel a base channel.

Procedure: Creating a custom channel

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and click **[Create Channel]**.
2. On the Create Software Channel page, give your channel a name (for example, My Tools SLES 15 SP1 x86_64) and a label (for example, my-

tools-sles15sp1-x86_64). Labels must not contain spaces or uppercase letters.

3. In the Parent Channel drop down, choose the relevant base channel (for example, SLE-Product-SLES15-SP1-Pool for x86_64). Ensure that you choose the compatible parent channel for your packages.
4. In the Architecture drop down, choose the appropriate hardware architecture (for example, x86_64).
5. Provide any additional information in the contact details, channel access control, and GPG fields, as required for your environment.
6. Click **[Create Channel]**.

Custom channels sometimes require additional security settings. Many third party vendors secure packages with GPG. If you want to use GPG-protected packages in your custom channel, you need to trust the GPG key which has been used to sign the metadata. You can then check the Has Signed Metadata? check box to match the package metadata against the trusted GPG keys.

If remote channels and repositories are signed with GPG keys, you can import and trust these GPG keys. For example, execute the spacewalk-repo-sync from the command line on the Uyuni Server:

```
/usr/bin/spacewalk-repo-sync -c <channellabelname> -t yum
```

This command and procedure is for temporary GPG key synchronization only. For storing the key permantenly, see this section later on.

The underlying zypper call will import the key, if it is available. The Web UI does not offer this feature.

This only works when the repository you want to mirror is set up in a special way and provides the "key" in the repository next to the signature. This is the case for all repositories generated by the Open Build Service (OBS). For other repositories special preparation steps are needed, as described further below.



By default, the Enable GPG Check field is checked when you create a new channel. If you would like to add custom packages and applications to your channel, make sure you uncheck this field to be able to install unsigned packages. Disabling the GPG check is a security risk if packages are from an untrusted source.

You can only add a repository to the Uyuni with the Web UI if it is a valid software repository. Check in advance that needed repository metadata are available. Tools such as createrepo and reprepro are useful in this regard. mgrpush can help with pushing a single RPM into a channel without creating a repository.

- For more information on createrepo_c see https://manpages.opensuse.org/Leap-15.6/createrepo_c/
- For more information on reprepro see https://manpages.opensuse.org/Leap-15.6/reprepro/

Procedure: Adding a software repository

1. In the Uyuni Web UI, navigate to **Software › Manage › Repositories**, and click **[Create Repository]**.
2. On the Create Repository page, give your repository a label (for example, my-tools-sles15sp1-x86_64-repo).
3. In the Repository URL field, provide the path to the directory that contains the repodata file (for example, file:///opt/mytools/). You can use any valid addressing protocol in this field.
4. Uncheck the Has Signed Metadata? check box.
5. OPTIONAL: Complete the SSL fields if your repository requires client certificate authentication.
6. Click **[Create Repository]**.

The above procedure only works if the repository you want to mirror provides the "key" in the repository next to the signature. This is the case for all repositories generated by the OBS, but it is typically not the case for repositories of operating systems that are not offered by the OBS.

If the repository you want to use does not have a GPG key you can provide one yourself and import the GPG key into the keyring manually. If you import the key into the /var/lib/spacewalk/gpgdir keyring using the gpg command line tool it would be stored permanently. The key would also persist if the chroot environment would be cleaned.

Procedure: Creating a software repository with GPG key

1. The command to import a key into the keyring, is:

```
mgrctl gpg upload /path/to/gpg.key
```

2. For more information, see **Client Configuration Guide › Autoinst Owngpgkey**.



Add Debian non-flat repositories using uyuni_suite, uyuni_component, and uyuni_arch query parameters.

uyuni_suite

is mandatory. In Debian documentation, this is also known as distribution. With this parameter you specify the apt source. Without this parameter the original approach is used. If the parameter ends with /, the repository is identified as flat.

uyuni_component

is optional. This parameter can specify only one component. It is not possible to list the components. An apt source entry allows to specify multiple components, but for Uyuni it is not possible. Instead you must create a separate repository for each component.

uyuni_arch

is optional. If omitted the architecture name is calculated with a SQL query for the channel from the database. Specify uyuni_arch explicitly if it does not match the architecture of the channel (sometimes architecture naming is ambiguous).

Here are some examples:

Table 2. Debian non-flat repository mapping

Type	Source line / URL
apt source line	deb https://pkg.jenkins.io/debian-stable binary/
URL mapping	https://pkg.jenkins.io/debian-stable?uyuni_suite=binary/
apt source line	deb https://deb.debian.org/debian/ stable main
URL mapping	https://deb.debian.org/debian/?uyuni_suite=stable&uyuni_component=main
apt source line	deb http://archive.ubuntu.com/ubuntu/ noble main
URL mapping	http://archive.ubuntu.com/ubuntu/?uyuni_suite=noble&uyuni_component=main



This following information about the Debian repository definition format is based on

https://wiki.debian.org/DebianRepository/Format#Overview.

The repository definition format is as follows:

```
deb uri suite [component1] [component2] [...]
```

For example:

```
deb https://deb.debian.org/debian/ stable main
```

or

```
deb https://pkg.jenkins.io/debian-stable binary/
```

For each pair of suite and component the specification defines a distinct URL calculated on the base URL + suite + component.

Procedure: Assigning the repository to a channel

1. Assign your new repository to your custom channel by navigating to **Software › Manage › Channels**, clicking the name of your newly created custom channel.
2. Navigate to the Repositories tab, and ensure the repository you want to assign to the channel is checked. Click **[Save Repositories]**.
3. By default, the synchronization process starts immediately.

For more information about channel synchronization, see **Administration Guide › Custom Channels › Custom Channel Sync**.

Procedure: Adding custom channels to an activation key

1. In the Uyuni Web UI, navigate to **Systems › Activation Keys**, and select the key you want to add the custom channel to.
2. On the Details tab, in the Child Channels listing, select the channel to associate. You can select multiple channels, if you need to.
3. Click **[Update Activation Key]**.

6.3.2. Custom channel synchronization

To avoid missing important updates, SUSE recommends to keep your custom channels up to date with the remote repositories changes.

By default, a synchronization will happen automatically for all custom channels you create. In particular, it will happen:

- after adding a repository to a channel from the UI or by using spacewalk-common-channels
- as part of the daily task mgr-sync-refresh-default, which will synchronize all your custom and vendor channels.

To disable this default behaviour, set in /etc/rhn/rhn.conf:

```
java.unify_custom_channel_management = 0
```

With this property turned off, no synchronization is performed automatically and, in order to keep a custom channel up to date, you need to:

- synchronize it manually by navigating to the Sync tab and click **[Sync Now]**,
- set up an automated synchronization schedule from the Repositories tab.

When the process is started, there are several ways to check if a channel has finished synchronizing:

- In the Uyuni Web UI, navigate to **Admin › Setup Wizard** and select the Products tab. This dialog displays a completion bar for each product when they are being synchronized.
- In the Uyuni Web UI, navigate to **Software › Manage › Channels**, then click the channel associated to the repository. Navigate to the menu:[Repositories > Sync] tab. The Sync Status is shown next to the repository name.
- Check the synchronization log file at the command prompt:

  ```
  tail -f /var/log/rhn/reposync/<channel-label>.log
  ```

  Each child channel generates its own log during the synchronization progress. You need to check all the base and child channel log files to be sure that the synchronization is complete.

The following custom channel synchronization options are available:

Retain packages in channels which have been removed from the repository

This turns off strict mode.

Do not sync errata

Do not synchronize patches.

Sync only latest packages

Synchronize latest package version only.

Create kickstartable tree

This option prepares a directory tree ready for Kickstart auto installation.

Terminate upon any error

Stop synchronizing if an error occurs.

These options will be saved persistently for each channel. The **[Sync now]** button also saves the channel options before performing the synchronization.

6.3.3. Add packages and patches to custom channels

When you create a new custom channel without cloning it from an existing channel, it does not contain any packages or patches. You can add the packages and patches you require using the Uyuni Web UI.

Custom channels can only include packages or patches that are cloned or custom, and they must match the base architecture of the channel. Patches added to custom channels must apply to a package that exists in the channel.

Procedure: Adding packages to custom channels

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and go to the Packages tab.
2. OPTIONAL: See all packages currently in the channel by navigating to the List/Remove tab.
3. Add new packages to the channel by navigating to the Add tab.
4. Select the parent channel to provide packages, and click **[View Packages]** to populate the list.
5. Check the packages to add to the custom channel, and click **[Add Packages]**.
6. When you are satisfied with the selection, click **[Confirm Addition]** to add the packages to the channel.

7. OPTIONAL: You can compare the packages in the current channel with those in a different channel by navigating to **Software › Manage › Channels**, and going to the **Packages › Compare** tab. To make the two channels the same, click the **[Merge Differences]** button, and resolve any conflicts.

Procedure: Adding patches to a custom channel

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and go to the Patches tab.
2. OPTIONAL: See all patches currently in the channel by navigating to the List/Remove tab.
3. Add new patches to the channel by navigating to the Add tab, and selecting what kind of patches you want to add.
4. Select the parent channel to provide patches, and click **[View Associated Patches]** to populate the list.
5. Check the patches to add to the custom channel, and click **[Confirm]**.
6. When you are satisfied with the selection, click **[Confirm]** to add the patches to the channel.

6.3.4. Manage custom channels

Uyuni administrators and channel administrators can alter or delete any channel.

To grant other users rights to alter or delete a channel, navigate to **Software › Manage › Channels** and select the channel you want to edit. Navigate to the Managers tab, and check the user to grant permissions. Click **[Update]** to save the changes.



If you delete a channel that has been assigned to a set of clients, it triggers an immediate update of the channel state for any clients associated with the deleted channel. This is to ensure that the changes are reflected accurately in the repository file.

You cannot delete Uyuni channels with the Web UI. Only custom channels can be deleted.

Procedure: Deleting custom channels

1. In the Uyuni Web UI, navigate to **Software › Manage › Channels**, and select the channel you want to delete.
2. Click **[Delete software channel]**.
3. On the Delete Channel page, check the details of the channel you are deleting, and check the Unsubscribe Systems checkbox to remove the custom channel from any systems that might still be subscribed.
4. Click **[Delete Channel]**.

When channels are deleted:

- the packages that are part of the deleted channel are not automatically removed. You are not able to update packages that have had their channel deleted.
- all activation keys referring to that channel as a base channel are also removed.

You can delete packages that are not associated with a channel in the Uyuni Web UI. Navigate to **Software › Manage › Packages**, check the packages to remove, and click **[Delete Packages]**.

6.4. Third-party Channels

Uyuni includes the ability to synchronize the content of optional third-party repositories with some of the products that can be managed by it.

Some third-party GPG keys are included by default in the Uyuni database, and some are not. For third-party keys that are not included, the relevant keys need to be imported when choosing to synchronize such third-party repositories.

GPG key management is performed through the Uyuni XML-RPC API. The mgrctl gpg subcommands read a key from a local file or remote URL and upload it to the server, so they can be run from any host that has mgrctl installed and network access to the server's API; they no longer require shell access into the server container.

Because the commands call the API, valid administrator credentials must be supplied. They can be provided on the command line with the --api-server, --api-user, and --api-password flags, through environment variables, or in the mgrctl configuration file.

To import a GPG key, use the following command syntax:

```
mgrctl gpg upload <path-to-gpg-key-file-or-URL>
```

The argument can be either a path to a local file containing an ASCII-armored GPG key, or an URL from which mgrctl can download the key from.

1. Example: Uploading a GPG key from a local file

```
mgrctl gpg upload repomd.xml.key
```

2. Example: Uploading a GPG key from a remote repository using a URL

```
mgrctl gpg upload https://<3rd-party-
domain>/path/to/repository/repodata/repomd.xml.key
```

To list the keys currently held in the Uyuni GPG keyring, run:

```
mgrctl gpg list
```

The output includes, for each key, its fingerprint, key type and size, and the associated user IDs. The fingerprint can be used to remove a key from the keyring:

```
mgrctl gpg remove <fingerprint>
```



When installing a package on the client, you may also need to trust the GPG key on the client itself. This requires that the GPG key is also available there.

For more information, see: **Client Configuration Guide › Gpg Keys**.

6.5. Configure Automatic Channels Synchronization

By default, channels are automatically synchronized with the upstream repositories to ensure that clients have access to the latest content and updates. However, in some cases, you may want to disable this automatic synchronization for specific channels.

This chapter explains how to disable automatic synchronization for a channel.

6.5.1. Verify channel synchronization status using the API

To check if a channel is set to be automatically synchronized you can call an API.

To check the synchronization status of a channel using the API, use the following command. Replace <Channel_Label> with the actual label of the channel you want to check

```
spacecmd api -- --args "<Channel_Label>" channel.software.isAutoSync
INFO: Connected to http://localhost/rpc/api as admin
```

```
[
  true
]
```

The command will return true if automatic synchronization is enabled, or false if it is disabled.

This information is also available in the Web UI. To check the synchronization status, navigate to the channel details page and look for the field Automatic Repository Synchronization.

6.5.2. Verify automatic channels synchronization status

You can also generate anoverview table for all configured software channels at once.

This script logs in, queries the channel.software.isAutoSync status for each channel retrieved by softwarechannel_list, and formats the output into a table.

```
spacecmd -q login

printf "%-70s | %s\n" "CHANNEL" "AUTO-SYNC"
printf "%s\n" "-----------------------------------------------------------------------
+-----------"

for I in $(spacecmd -q softwarechannel_list) ; do
    STATUS=$(spacecmd -q api -- --args "$I" channel.software.isAutoSync | tr -d '[] \n\r')
    printf "%-70s | %s\n" "$I" "$STATUS"
done
```



This loop will also print the synchronization status for Content Lifecycle Management (CLM) channels.

6.5.3. Set a channel to skip automatic synchronization

To enable or disable automatic synchronization for a channel, you can use the API call channel.software.setAutoSync.

Here is an example of how to set a channel to skip automatic synchronization using the API (replace <Channel_Label> with the actual label of the channel you want to update):

```
spacecmd api -- --args '["<Channel_Label>", false]' channel.software.setAutoSync
INFO: Connected to http://localhost/rpc/api as admin
[
  false
]
```

To re-enable automatic synchronization, set the second parameter to true:

```
spacecmd api -- --args '["<Channel_Label>", true]' channel.software.setAutoSync
INFO: Connected to http://localhost/rpc/api as admin
```

```
[
  true
]
```

6.5.4. Manually synchronize a channel

Even if you have disabled automatic synchronization for a channel, you can still manually synchronize it whenever needed.



Use mgrctl term before running steps inside the server container.

To manually synchronize a channel, you can use the command(replace <Channel_Label> with the actual label of the channel you want to synchronize):

```
spacewalk-repo-sync -c <Channel_Label>
```

Chapter 7. Confidential Computing

Confidential Computing is a technology which allows protection of data in use by using hardware-based Trusted Execution Environment (TEE), the type of environments that provide increased level of security for data integrity, data confidentiality, and code integrity.

7.1. Confidential Computing with Uyuni

The trustworthiness of the TEE is checked with the attestation process. Uyuni can be used as an attestation server for the systems registered to it. It generates a report page for the systems which run in this mode. These systems need to be attested and checked on regular base. The history of the past checks is also stored and available per request.

Confidential Computing Attestation depends on the used hardware and environment where the attested systems are running on.



Confidential Computing Attestation is only available when Uyuni is running on x86-64 architecture.

7.2. Requirements

Confidential Computing can be set up in an environment with the following characteristics:

- Attested system (virtual machine) is bootstrapped to Uyuni and runs on one of the following systems:
  - SUSE Linux Enterprise 15 SP6
  - SUSE Linux Enterprise 15 SP7
  - SUSE Linux Enterprise 16
  - openSUSE Leap 15 SP6
  - openSUSE Leap 15 SP7
  - openSUSE Leap 16
- Hardware must have one of the following CPUs:
  - AMD EPYC Milan
  - AMD EPYC Genoa
  - AMD EPYC Bergamo
  - AMD EPYC Siena
  - AMD EPYC Turin
  - IBM z16

- IBM z17

- BIOS must be configured to allow Confidential Computing attestation
- Host OS and the virtualization software (KVM and libvirt) must support Confidential Computing
- For information about required URLs when performing confidential computing attestation for IBM, see see **Installation and Upgrade Guide › Required URLs**.

7.3. Limitations

Secure boot is attested. However, currently KVM secure boot and AMD attestation tool snpguest are not working together.

7.4. Use Confidential Computing in Uyuni



For the exact steps for setting up and configuring Confidential Computing on your host, refer to the OS Vendor documentation.

Procedure: Enabling Attestation Container During the Uyuni Installation

1. The attestation container is enabled during the installation of Uyuni with mgradm install.
2. Add the following to file mgradm.yaml.

```
coco:
    replicas: 1
```

Procedure: Enabling Attestation Container After the Uyuni Installation

1. To enable the attestation container after the installation, use the command line parameter mgradm.
2. Run the command

```
mgradm scale uyuni-server-attestation --replicas 1
```

Procedure: Disabling Attestation Container After the Uyuni Installation

1. To disable the already enabled attestation container, run the command:

```
mgradm scale uyuni-server-attestation --replicas 0
```

Procedure: Enabling Attestation

1. For the selected system, go to tab **Audit › Confidential Computing › Settings**.
2. Enable the attestation by selecting the toggle button.

3. In the field Environment Type select the correct option from the drop-down list.
4. If the field Environment Type is set to one of the IBM z-series CPU values, two more settings are required, Host Key Document and Secure Execution Header.



Host Key Document is a public key certificate of the IBM hypervisor server instance where the attested system is going to run.

Secure Execution Header is a binary file needed to build the IBM secure VM image of the attested system.

Both files should be requested by the administrator creating the attested system VM running on the IBM hypervisor.



Host Key Document and Secure Execution Header are not security sensitive, therefore it is safe to store them on the Uyuni server.

5. Click button **[Save]** to save the changes.

Procedure: Scheduling New Attestation

1. For the selected system, go to tab **Audit › Confidential Computing › List Attestations**.
2. Click **[Schedule Attestation]**. The new form opens.
3. In the field Earliest select the time of running the attestation.
4. If needed, add the newly created attestation to the action chain by selecting Add to option.
5. Click button **[Schedule]** to save and schedule the new attestation execution.

Procedure: Viewing Attestation Reports from System Details

1. For the selected system, go to tab **Audit › Confidential Computing › List Attestations**.
2. Find and select the report you want to view.
3. After clicking the selected attestation report tab Overview will open.
    a. If the selected CPU is AMD you will be shown additional tabs SEV-SNP and Secure boot.
    b. If the selected CPU is IBM you will be shown additional tab IBM SELPVATTEST.

Procedure: Viewing Attestation Reports from Audit

1. From the navigation bar, select **Audit › Confidential Computing**.
2. The list of all attestations will be shown in the main panel.
3. Find and select the report you want to view.

7.4.1. Report Statuses

Attestation reports can have one of the following statuses:

Pending

This is the default status of the scheduled attestation. The report is still not available, either because the process has not yet started or completed.

Successful

When the scheduled attestation creates a report which can be viewed, the status of the process is Successful.

Failed

When the scheduled fails and does not create a report as a result, the status of the process is Failed.

7.5. Related Topics

For more information about Confidential Computing, see here.

Chapter 8. Content Lifecycle Management

Content lifecycle management allows you to customize and test packages before updating production clients. This is especially useful if you need to apply updates during a limited maintenance window.

Content lifecycle management allows you to select software channels as sources, adjust them as required for your environment, and thoroughly test them before installing onto your production clients.

While you cannot directly modify vendor channels, you can clone them and then modify the clones by adding or removing packages and custom patches. You can assign these cloned channels to test clients to ensure they work as expected.



By default, cloned vendor channels match the original vendor channel and automatically select the dependencies. You can disable the automatic selection for cloned channels by adding the following option in /etc/rhn/rhn.conf:

```
java.cloned_channel_auto_selection = false
```

Then, when all tests pass, you can promote the cloned channels to production servers.

This is achieved through a series of environments that your software channels can move through on their lifecycle. Most environment lifecycles include at least test and production environments, but you can have as many environments as you require.

This section covers the basic content lifecycle procedures, and the filters available. For more specific examples, see **Administration Guide › Content Lifecycle Examples**.

8.1. Create a content lifecycle project

To set up a content lifecycle, you need to begin with a project. The project defines the software channel sources, the filters used to find packages, and the build environments.

Procedure: Creating a content lifecycle project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and click **[Create Project]**.
2. In the Label field, enter a label for your project. The Label field only accepts lowercase letters, numbers, periods, hyphens, and underscores.
3. In the Name field, enter a descriptive name for your project.
4. Click the **[Create]** button to create your project and return to the project

page.

5. Click **[Attach/Detach Sources]**.
6. In the Sources dialog, select the source type, and select a base channel for your project. The available child channels for the selected base channel are displayed, including information on whether the channel is mandatory or recommended.
7. Check the child channels you require, and click **[Save]** to return to the project page. The software channels you selected should now be showing.
8. Click **[Attach/Detach Filters]**.
9. In the Filters dialog, select the filters you want to attach to the project. To create a new filter, click **[Create new Filter]**.
10. Click **[Add Environment]**.
11. In the Environment Lifecycle dialog, give the first environment a name, a label, and a description, and click **[Save]**. The Label field only accepts lowercase letters, numbers, periods, hyphens, and underscores.
12. Continue creating environments until you have all the environments for your lifecycle completed. You can select the order of the environments in the lifecycle by selecting an environment in the Insert before field when you create it.

8.2. Filter types

Uyuni allows you to create various types of filters to control the content used for building the project. Filters allow you to select which packages are included or excluded from the build. For example, you could exclude all kernel packages, or include only specific releases of some packages.

The supported filters are:

- package filtering
  - by name
  - by name, epoch, version, release, and architecture

    - by provided name
- patch filtering
    - by advisory name
    - by advisory type
    - by synopsis
    - by keyword
    - by date
    - by affected package
- module
    - by stream



Package dependencies are not resolved during content filtering.

There are multiple matchers you can use with the filter. Which ones are available depends on the filter type you choose.

The available matchers are:

- contains
- matches (must take the form of a regular expression)
- equals
- greater
- greater or equal
- lower or equal
- lower
- later or equal

8.2.1. Filter rule parameter

Each filter has a rule parameter that can be set to either Allow or Deny. The filters are processed like this:

- If a package or patch satisfies a Deny filter, it is excluded from the result.
- If a package or patch satisfies an Allow filter, it is included in the result (even if it was excluded by a Deny filter). Allow filters for packages only operate on package filters and Allow filters for patches only operate on patch filters. This means with a package filter you cannot add packages back in that were filtered out through patch filters or vice versa for patches that where filtered out through package filters.

This behavior is useful when you want to exclude large number of packages or patches using a general Deny filter and "cherry-pick" specific packages or patches with specific Allow filters.

Content filters are global in your organization and can be shared between projects.

If your project already contains built sources, when you add an environment it is automatically populated with the existing content. Content is drawn from the previous environment of the cycle if it had one. If there is no previous environment, it is left empty until the project sources are built again.

8.3. Filter templates

To help with creating filters for some common scenarios, Uyuni offers filter templates. When applied, these templates help creating a set of filters in advance, tailored for a specific use case.

This section describes available templates and their usages.

8.3.1. Live patching based on a SUSE product

In a project that contains live patching, regular future kernel packages must be excluded so that only live patch packages are offered as updates to clients. On the other hand, already installed regular kernel packages must still be included to keep system integrity.

When applied, this template creates three filters required to achieve this behavior:

- Allow patches that contain kernel-default package equal to a base kernel version
- Deny patches that contain reboot_suggested keyword
- Deny patches that contain a package which provides the name installhint(reboot-needed)

For more information on how to set up a live patching project, see **Administration Guide › Content Lifecycle Examples › Exclude Higher Kernel Version**.

Procedure: Applying the template

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters**, and click **[Create Filter]**.
2. In the dialog, click **[Use a template]**. The inputs will change accordingly.
3. In the Prefix field, type a name prefix. This value will be prepended to the name of every individual filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the

project label.

4. In the Template field, select Live patching based on a SUSE product.
5. In the Product field, select the product you wish to set up live patching for.
6. In the Kernel field, select a kernel version from the list of versions available in the selected product. The filter to deny the later regular kernel patches will be based on this version.
7. Click [Save] to create the filters.
8. Navigate to **Content Lifecycle › Projects** and select your project.
9. Click [Attach/Detach Filters].
10. Select the three filters that have the specified prefix, and click [Save].

8.3.2. Live patching based on a system

When you want to set up a live patching project based on a kernel version installed in a specific registered system, you can use the "live patching based on a system" template.

When applied, this template creates three filters required to achieve this behavior:

- Allow patches that contain kernel-default package equal to a base kernel version
- Deny patches that contain reboot_suggested keyword
- Deny patches that contain a package which provides the name installhint(reboot-needed)

For more information on how to set up a live patching project, see **Administration Guide › Content Lifecycle Examples › Exclude Higher Kernel Version**.

Procedure: Applying the template

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters**, and click [Create Filter].
2. In the dialog, click [Use a template]. The inputs will change accordingly.
3. In the Prefix field, type a name prefix. This value will be prepended to the name of every filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the

project label.

4. In the Template field, select Live patching based on a specific system.
5. In the System field, select a system from the list, or start typing a system name to narrow down the options.
6. In the Kernel field, select a kernel version from the list of versions installed in the selected system. The filter to deny the later regular kernel patches will be based on this version.
7. Click **[Save]** to create the filters.
8. Navigate to **Content Lifecycle › Projects** and select your project.
9. Click **[Attach/Detach Filters]**.
10. Select the three filters that have the specified prefix, and click **[Save]**.

8.3.3. AppStream modules with defaults

When you want to have all the modules available in a modular repository included in your project, you can automatically add them using this filter template.

When applied, this template creates an AppStream filter per module and its default stream.

If this process is done from the project's page, the filters are added to the project automatically. Otherwise, the created filters can be listed in **Content Lifecycle › Filters** and be added to any project as needed.

Each individual filter can be edited to select a different module stream, or removed altogether to exclude that module from the target repositories.



Because not all module streams are compatible with each other, changing individual streams may prevent successful resolution of modular dependencies. When this happens, the filters pane in the project details page will show an error describing the problem, and the build button will be disabled until all the module selections are compatible.



Since Red Hat Enterprise Linux 9, modules do not have any defined default streams. Therefore, using this template with Red Hat Enterprise Linux 9 sources will have no effect.

For more information on how to set up AppStream repositories with content lifecycle management, see **Administration Guide › Content Lifecycle Examples › Appstream Filters**.

Procedure: Applying the template

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and select your project.
2. In the Filters section, click **[Attach/Detach Filters]**, and then click **[Create New Filter]**.
3. In the dialog, click **[Use a template]**. The inputs will change accordingly.
4. In the Prefix field, type a name prefix. This value will be prepended to the name of every filter created by the template. If the template is being applied in the context of a project, this field will be prefilled with the project label.
5. In the Template field, select AppStream modules with defaults.
6. In the Channel field, select a modular channel to get the modules from. In this dropdown, only the modular channels are displayed.
7. Click **[Save]** to create the filters.
8. Scroll to the Filters section to see the newly attached AppStream filters.
9. You can edit/remove any individual filter to tailor the project to your needs.

8.4. Build a content lifecycle project

When you have created your project, defined environments, and attached sources and filters, you can build the project for the first time.

Building applies filters to the attached sources and clones them to the first environment in the project.

You can use the same vendor channels as sources for multiple content projects. In this case, Uyuni does not create new patch clones for each cloned channel. Instead, a single patch clone is shared between all of your cloned channels. This can cause problems if a vendor modifies a patch; for example, if the patch is retracted, or the packages within the patch are changed. When you build one of the content projects, all the channels that share the cloned patch are synchronized with the original by default, even if the channels are in other environments of your content project, or other content project channels in your organization. You can change this behavior by turning off automatic patch synchronization in your organization settings. To manually synchronize the patch later for all channels sharing the patch, navigate to **Software › Manage › Channels**, click

the channel you want to synchronize and navigate to the Sync subtab. Even manual patch synchronization affects all organization channels sharing the patch.

Procedure: Building a content lifecycle project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and select the project you want to build.



Make sure you have the environment available before building the project.

2. Review the attached sources and filters, and click **[Build]**.
3. Provide a version message to describe the changes or updates in this build.
4. You can monitor build progress in the Environment Lifecycle section.

After the build is finished, the environment version is increased by one and the built sources, such as software channels, can be assigned to your clients.

8.5. Promote environments

When the project has been built, the built sources can be sequentially promoted to the environments.

Procedure: Promoting environments

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects**, and select the project you want to work with.
2. In the Environment Lifecycle section, locate the environment to promote to its successor, and click **[Promote]**.
3. You can monitor build progress in the Environment Lifecycle section.

8.6. Assign clients to environments

When you build and promote content lifecycle projects, Uyuni creates a tree of software channels. To add clients to the environment, assign the base and child software channels to your client using **Software › Software Channels** in the System Details page for the client.



Newly added cloned channels are not assigned to clients automatically. If you add or promote sources you need to manually check and update your channel assignments.

> Automatic assignment is intended to be added to Uyuni in a future version.

8.7. Content Lifecycle Management Examples

This section contains some common examples of how you can use content lifecycle management. Use these examples to build your own personalized implementation.

8.7.1. Creating a Project for a Monthly Patch Cycle

An example project for a monthly patch cycle consists of:

- Creating a By Date filter
- Adding the filter to the project
- Applying the filter to a new project build
- Excluding a patch from the project
- Including a patch in the project

8.7.1.1. Creating a By Date filter

The By Date filter excludes all patches released after a specified date. This filter is useful for your content lifecycle projects that follow a monthly patch cycle.

Procedure: Creating the By Date filter

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click **[Create Filter]**.
2. In the Filter Name field, type a name for your filter. For example, Exclude patches by date.
3. In the Filter Type field, select Patch (Issue date).
4. In the Matcher field, later or equal is autoselected.
5. Select the date and time.
6. Click **[Save]**.

8.7.1.2. Add a filter to the project

Procedure: Adding a filter to a project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Projects** and select a project from the list.
2. Click **[Attach/Detach Filters]** link to see all available filters
3. Select the new Exclude patches by date filter.
4. Click **[Save]**.

8.7.1.3. Apply a filter to a new project build

The new filter is added to your filter list, but it still needs to be applied to the project. To apply the filter you need to build the first environment.

Procedure: Using the filter

1. Click **[Build]** to build the first environment.
2. OPTIONAL: Add a message. You can use messages to help track the build history.
3. Check that the filter has worked correctly by using the new channels on a test server.
4. Click **[Promote]** to move the content to the next environment. The build takes longer if you have a large number of filters, or they are very complex.

8.7.1.4. Exclude a patch from the project

Tests may help you discover issues. When an issue is found, exclude the problem patch released before the by date filter.

Procedure: Excluding a patch

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click **[Create Filter]**.
2. In the Filter Name field, enter a name for the filter. For example, Exclude openjdk patch.

3. In the Filter Type field, select Patch (Advisory Name).
4. In the Matcher field, select equals.
5. In the Advisory Name field, type a name for the advisory. For example, SUSE-15-2019-1807.
6. Click [Save].
7. Navigate to **Content Lifecycle › Projects** and select your project.
8. Click [Attach/Detach Filters] link, select Exclude openjdk patch, and click [Save].

When you rebuild the project with the [Build] button, the new filter is used together with the by date filter we added before.

8.7.1.5. Include a patch in the project

In this example, you have received a security alert. An important security patch was released several days after the first of the month you are currently working on. The name of the new patch is SUSE-15-2019-2071. You need to include this new patch into your environment.



The Allow filters rule overrides the exclude function of the Deny filter rule. For more information, see **Administration Guide › Content Lifecycle**.

Procedure: Including a patch in a project

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters** and click [Create Filter].
2. In the Filter Name field, type a name for the filter. For example, Include kernel security fix.
3. In the Filter Type field, select Patch (Advisory Name).
4. In the Matcher field, select equals.
5. In the Advisory Name field, type SUSE-15-2019-2071, and check Allow.
6. Click [Save] to store the filter.
7. Navigate to **Content Lifecycle › Projects** and select your project from the list.

8. Click [Attach/Detach Filters], and select Include kernel security patch.
9. Click [Save].
10. Click [Build] to rebuild the environment.

8.7.2. Update an existing monthly patch cycle

When a monthly patch cycle is complete, you can update the patch cycle for the next month.

Procedure: Updating a monthly patch cycle

1. In the by date field, change the date of the filter to the next month. Alternatively, create a new filter and change the assignment to the project.
2. Check if the exclude filter for SUSE-15-2019-1807 can be detached from the project. There may be a new patch available to fix this issue.
3. Detach the allow filter you added previously. The patch is included by default.
4. Rebuild the project to create a new environment with patches for the next month.

8.7.3. Enhance a project with live patching

This section covers setting up filters to create environments for live patching.

When you are preparing to use live patching, there are some important considerations:

- Only ever use one kernel version on your systems. The live patching packages are installed with a specific kernel.
- Live patching updates are shipped as one patch.
- Each kernel patch that begins a new series of live patching kernels displays the required reboot flag. These kernel patches come with live patching tools. When you have installed them, you need to reboot the system at least once before the next year.
- Only install live patch updates that match the installed kernel version.
- Live patches are provided as stand-alone patches. You must exclude all regular kernel patches with higher kernel version than the currently installed one.

8.7.3.1. Exclude packages with a higher kernel version

In this example you update your systems with the SUSE-15-2019-1244 patch. This patch contains kernel-default-4.12.14-150.17.1-x86_64.

You need to exclude all patches which contain a higher version of kernel-default and kernel-default-base.

Procedure: Excluding packages with a higher kernel version

1. In the Uyuni Web UI, navigate to **Content Lifecycle › Filters**, and click **[Create Filter]**.
2. In the Filter Name field, type a name for your filter. For example, Exclude kernel greater than 4.12.14-150.17.1.
3. In the Filter Type field, select Patch (Contains Package).
4. In the Matcher field, select version greater than.
5. In the Package Name field, type kernel-default.
6. Leave the Epoch field empty.
7. In the Version field, type 4.12.14.
8. In the Release field, type 150.17.1.
9. Click **[Save]** to store the filter.
10. Navigate to **Content Lifecycle › Projects** and select your project.
11. Click **[Attach/Detach Filters]**.
12. Select Exclude kernel greater than 4.12.14-150.17.1, and click **[Save]**.

You need to repeat this procedure for the kernel-default-base package.

When you click **[Build]**, a new environment is created. The new environment contains all the kernel patches up to the version you installed.



All kernel patches with higher kernel versions are removed. Live patching kernels remain available as long as they are not the first in a series.

This procedure can be automated using a filter template. For more information on how to apply a live patching filter template, see **Administration Guide › Content Lifecycle › Filter Templates**.

8.7.4. Switch to a new kernel version for live patching

Live patching for a specific kernel version is only available for one year. After one year you must update the kernel on your systems. Execute these environment changes:

Procedure: Switch to a new kernel version

1. Decide which kernel version to upgrade to. For example: 4.12.14-150.32.1
2. Create a new kernel version Filter.
3. Detach all previous Live Patching filters associated with the old kernel.
4. Attach the new kernel version filter.
5. Click **[Build]** to rebuild the environment.



Once the build is complete, you must immediately re-attach the two filters which:

- Deny patches that contain reboot_suggested keyword
- Deny patches that contain a package which provides the name installhint(reboot-needed)

The new environment contains all kernel patches up to the new kernel version you selected. You need to reboot systems after they have performed the upgrade. The new kernel remains valid for one year. All packages installed during the year match the current live patching kernel filter.

8.7.5. AppStream filters

In a content lifecycle management project, you can use the AppStream filter to transform modular repositories into regular repositories. It does this by keeping the packages in the repository and stripping away the module metadata. The resulting repository can be used in Uyuni in the same way as a regular repository.



AppStream repositories are natively supported throughout the Web UI.

Therefore, this process is not mandatory to work with AppStream repositories.

The AppStream filter selects a single module stream to be included in the target repository. You can add multiple filters to select multiple module streams.

If you do not use an AppStream filter in your CLM project, the module metadata in the modular sources remains intact, and the target repositories contain the same module metadata. As long as at least one

AppStream filter is enabled in the CLM project, all target repositories are transformed into regular repositories.

In some cases, you might wish to build regular repositories without having to include packages from any module. To do so, add an AppStream filter using the matcher none (disable modularity). This will disable all the modules in the target repository. This is especially useful for Red Hat Enterprise Linux 9 clients, where the default versions of most modules are already included in the AppStream repository as regular packages.

To use the AppStream filter, you need a CLM project with a modular repository such as Red Hat Enterprise Linux AppStreams. Ensure that you included the module you need as a source before you begin.

Procedure: Using AppStream filters

1. In the Uyuni Web UI, navigate to your Red Hat Enterprise Linux 8 or 9 CLM project. Ensure that you have included the AppStream channels for your project.
2. Click **[Create Filter]** and use these parameters:
    - In the Filter Name field, type a name for the new filter.
    - In the Filter Type field, select Module (Stream).
    - In the Matcher field, select equals.
    - In the Module Name field, type a module name. For example, postgresql.
    - In the Stream field, type the name of the desired stream. For example, 10. If you leave this field blank, the default stream for the module is selected.
3. Click **[Save]** to create the new filter.
4. Navigate to **Content Lifecycle › Projects** and select your project.
5. Click **[Attach/Detach Filters]**, select your new AppStream filter, and click **[Save]**.

You can use the browse function in the Create/Edit Filter form to select a module from a list of available module streams for a modular channel.

Procedure: Browsing available module streams

1. In the Uyuni Web UI, navigate to your Red Hat Enterprise Linux 8 or 9 CLM project. Ensure that you have included the AppStream channels for your

project.

2. Click **[Create Filter]** and use these parameters:
    - In the Filter Name field, type a name for the new filter.
    - In the Filter Type field, select Module (Stream).
    - In the Matcher field, select equals.
3. Click Browse available modules to see all modular channels.
4. Select a channel to browse the modules and streams:
    - In the Module Name field, start typing a module name to search, or select from the list.
    - In the Stream field, start typing a stream name to search, or select from the list.



Channel selection is only for browsing modules. The selected channel is not be saved with the filter, and does not affect the CLM process in any way.

You can create additional AppStream filters for any other module stream to be included in the target repository. Any module streams that the selected stream depends on is automatically included.



Be careful not to specify conflicting, incompatible, or missing module streams. For example, selecting two streams from the same module is invalid.

Procedure: Disabling modularity

1. In the Uyuni Web UI, navigate to your Red Hat Enterprise Linux 8 or 9 CLM project. Ensure that you have included the AppStream channels for your project.
2. Click **[Create Filter]** and use these parameters:
    - In the Filter Name field, type a name for the new filter.
    - In the Filter Type field, select Module (Stream).
    - In the Matcher field, select none (disable modularity).
3. Click **[Save]** to create the new filter.

4. Navigate to **Content Lifecycle › Projects** and select your project.
5. Click **[Attach/Detach Filters]**, select your new AppStream filter, and click **[Save]**.

This will effectively remove the module metadata from the target repository, excluding any package that belongs to a module.

When you build your CLM project using the **[Build]** button in the Web UI, the target repository is a regular repository without any modules, that contains packages from the selected module streams.



Disabling modularity altogether in Red Hat Enterprise Linux 8 projects might result in a faulty environment as some modules are essential for healthy operation in Red Hat Enterprise Linux 8.

Chapter 9. Content Staging

Staging is used by clients to download packages in advance, before they are installed. This allows package installation to begin as soon as it is scheduled, which can reduce the amount of time required for a maintenance window.

9.1. Enable Content Staging

You can manage content staging across your entire organization. In the Uyuni Web UI, navigate to **Admin › Organizations** to see a list of available organizations. Click the name of an organization, and check the Enable Staging Contents box to allow clients in this organization to stage package data.



You must be logged in as the Uyuni administrator to create and manage organizations.

You can also enable staging at the command prompt by editing /etc/sysconfig/rhn/up2date, and adding or editing these lines:

```
stagingContent=1
stagingContentWindow=24
```

The stagingContentWindow parameter is a time value expressed in hours and determines when downloading starts. It is the number of hours before the scheduled installation or update time. In this example, content is downloaded 24 hours before the installation time. The start time for download depends on the selected contact method for a system.

Next time an action is scheduled, packages are automatically downloaded, but not installed. At the scheduled time, the staged packages are installed.

9.2. Configure Content Staging

There are two parameters used to configure content staging:

- salt_content_staging_advance is the advance time for the content staging window to open, in hours. This is the number of hours before installation starts, that package downloads can begin.
- salt_content_staging_window is the duration of the content staging window, in hours. This is the amount of time clients have to stage packages before installation begins.

For example, if salt_content_staging_advance is set to six hours, and salt_content_staging_window is set to two hours, the staging window opens six hours before the installation time, and remain open for two hours. No packages are downloaded in the four remaining hours until installation starts.

If you set the same value for both salt_content_staging_advance and salt_content_staging_window packages are able to be downloaded until installation begins.

Configure the content staging parameters in /usr/share/rhn/config-defaults/rhn_java.conf.

Default values:

- salt_content_staging_advance: 8 hours
- salt_content_staging_window: 8 hours



Content staging must be enabled for these parameters to work correctly.

Chapter 10. Disconnected Setup

When it is not possible to connect Uyuni to the internet, you can use it within a disconnected environment.

The repository mirroring tool (RMT) is available on SUSE Linux Enterprise 15 and later. RMT replaces the subscription management tool (SMT), which can be used on older SUSE Linux Enterprise installations.

In a disconnected Uyuni setup, RMT or SMT uses an external network to connect to SUSE Customer Center. All software channels and repositories are synchronized to a removable storage device. The storage device can then be used to update the disconnected Uyuni installation.

This setup allows your Uyuni installation to remain in an offline, disconnected environment.



Your RMT or SMT instance must be used to manage the Uyuni Server directly. It cannot be used to manage a second RMT or SMT instance, in a cascade.

For more information on RMT, see https://documentation.suse.com/sles/15-SP7/html/SLES-all/book-rmt.html.

10.1. Synchronize Channels and Repositories from SCC

10.1.1. Synchronize RMT

You can use RMT on SUSE Linux Enterprise 15 installations to manage clients running SUSE Linux Enterprise 12 or later.

We recommend you set up a dedicated RMT instance for each Uyuni installation.

Procedure: Setting up RMT

1. On the RMT instance, install the RMT package:

```
zypper in rmt-server
```

2. Configure RMT using YaST:

```
yast2 rmt
```

3. Follow the prompts to complete installation.

For more information on setting up RMT, see https://documentation.suse.com/sles/15-SP7/html/SLES-all/book-rmt.html.

Procedure: Synchronizing RMT with SCC

1. On the RMT instance, list all available products and repositories for your organization:

```
rmt-cli products list --all
rmt-cli repos list --all
```

2. Synchronize all available updates for your organization:

```
rmt-cli sync
```

   You can also configure RMT to synchronize regularly using systemd.

3. Enable the products you require. For example, to enable SLES 15:

```
rmt-cli product enable sles/15/x86_64
```

4. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at /mnt/usb:

```
rmt-cli export data /mnt/usb
```

5. Export the enabled repositories to your removable storage:

```
rmt-cli export settings /mnt/usb
rmt-cli export repos /mnt/usb
```



Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change RMT user settings in the cli section of /etc/rmt.conf.

10.1.2. Synchronize SMT

SMT is included with SUSE Linux Enterprise 12, and can be used to manage clients running SUSE Linux Enterprise 10 or later.

SMT requires you to create a local mirror directory on the SMT instance to synchronize repositories and packages.

For more details on installing and configuring SMT, see https://documentation.suse.com/sles/12-SP5/html/SLES-all/book-smt.html.

Procedure: Synchronizing SMT with SCC

1. On the SMT instance, create a database replacement file:

```
smt-sync --createdbreplacementfile /tmp/dbrepl.xml
```

2. Export the synchronized data to your removable storage. In this example, the storage medium is mounted at /mnt/usb:

```
smt-sync --todir /mnt/usb
smt-mirror --dbreplfile /tmp/dbrepl.xml --directory /mnt/usb \
           --fromlocalsmt -L /var/log/smt/smt-mirror-export.log
curl https://scc.suse.com/multi-linux-manager/product_tree.json -o
/mnt/usb/product_tree.json
```



Ensure that the external storage is mounted to a directory that is writeable by the RMT user. You can change SMT user settings in /etc/smt.conf.

10.2. Mandatory Channels

The corresponding Uyuni Client Tools Channels are required, for Uyuni to be able to synchronize a given channel. If these channels are not enabled, Uyuni may fail to detect that product.

Run the following command to enable these mandatory channels:

SLES 12 and products based on it such as SLES for SAP or SLE HPC

RMT: rmt-cli products enable sle-manager-tools/12/x86_64

SMT: smt repos -p sle-manager-tools,12,x86_64

SLES 15 and products based on it such as SLES for SAP or SLE HPC

RMT: rmt-cli products enable sle-manager-tools/15/x86_64

SMT: smt repos -p sle-manager-tools,15,x86_64

Then mirror the channels, and export.

Other distributions, or architectures, can be enabled. For more information about enabling product channels or repositories to be mirrored, see the documentation:

RMT

https://documentation.suse.com/sles/15-SP7/html/SLES-all/cha-rmt-mirroring.html#sec-rmt-mirroring-enable-disable

SMT

https://documentation.suse.com/sles/12-SP5/single-html/SLES-smt/index.html#smt-

mirroring-manage-domirror

10.3. Disconnected Server

To set up Uyuni as a disconnected server follow the instructions of an air-gapped deployment.

10.3.1. Deploy

It is recommended to deploy a disconnected server as a Virtual Machine (VM) using a provided image. For an air-gapped deployment of Uyuni Server, see **Installation and Upgrade Guide › Container Deployment › Server Air Gapped Deployment Mlm**.

Keep in mind to execute the final command with the --mirror option and replace </media/disk> with your mount point:

```
mgradm install --mirror </media/disk>
```

10.3.2. Synchronize

When you have removable media loaded with your SUSE Customer Center data, you can use it to synchronize your disconnected server.



The removable media that you use for synchronization must always be available at the same mount point. Do not trigger a synchronization if the storage medium is not mounted. This results in data corruption.

Procedure: Synchronizing a Disconnected Server

1. Restart the Tomcat service:

```
mgrctl exec -ti -- systemctl restart tomcat
```

2. Refresh the local data:

```
mgrctl exec -ti -- mgr-sync refresh
```

3. Perform a synchronization:

```
mgrctl exec -ti -- mgr-sync list channels
mgrctl exec -ti -- mgr-sync add channel channel-label
```



Be aware that if server.susemanager.fromdir is set, Uyuni will not be able to check if SUSE Customer Center credentials are valid or not. Instead, a warning sign will be displayed and

> no SCC online check will be performed.

An alternative to disconnected setup may be to copy content between servers using Inter-Server Synchronization (ISS). For more information, see **Specialized Guides › Large Deployments › Inter-Server Synchronization - Version 2**.

Chapter 11. Disk Space Management

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in some cases, is not recoverable.

Uyuni monitors some directories for free disk space. You can modify which directories are monitored, and the warnings that are created using environment variables in systemd service override files.

When the available space in one of the monitored directories falls below a warning threshold, a message is sent to the configured email address and a notification is shown at the top of the sign-in page.

Only the file systems that hold the monitored directories are checked.

11.1. Monitored Directories

By default, the Uyuni Server container monitors these directories:

- /var/spacewalk
- /var/cache
- /srv

You can change which directories are monitored by setting the PODMAN_EXTRA_ARGS environment variable in the server container systemd service override file, which passes DISKCHECKDIRS to the container. You can specify multiple directories by separating them with a space.



The DISKCHECKDIRS environment variable applies only to the server container. The database container always monitors /var/lib/pgsql/data and this cannot be overridden.

Procedure: Configuring Monitored Directories

1. Create a systemd override directory for the server service:

```
mkdir -p /etc/systemd/system/uyuni-server.service.d/
```

2. Create a configuration file with the custom directories:

```
cat > /etc/systemd/system/uyuni-server.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKDIRS=/var/spacewalk /var/cache /srv"
EOF
```

3. Apply the configuration by restarting Uyuni:

```
mgradm restart
```

For more information about container volumes, see **Installation and Upgrade Guide › Container Management › Persistent Container Volumes**.

11.2. Thresholds

By default, Uyuni creates a warning when a monitored directory reaches 90% disk usage. A critical alert is created when a monitored directory reaches 95% disk usage, and the container health check will fail.

You can change these alert thresholds by setting the DISKCHECKALERT and DISKTHRESHOLD environment variables.

Procedure: Configuring Disk Check Thresholds

1. Create systemd override directories:

```
mkdir -p /etc/systemd/system/uyuni-server.service.d/
mkdir -p /etc/systemd/system/uyuni-db.service.d/
```

2. Create configuration files for both server and database services:

```
cat > /etc/systemd/system/uyuni-server.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT=90 --env DISKTHRESHOLD=95"
EOF

cat > /etc/systemd/system/uyuni-db.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT=90 --env DISKTHRESHOLD=95"
EOF
```

3. Apply the configuration by restarting Uyuni:

```
mgradm restart
```



For large deployments with multi-terabyte storage volumes, you may want to use higher thresholds to make more efficient use of disk space. For example, on a 10 TB volume, keeping 5% free means 500 GB unused. For tuning recommendations for large deployments, see **Specialized Guides › Large Deployments › Tuning › Diskcheck Thresholds**.

11.2.1. Free space thresholds

On large volumes, a percentage can raise alerts while plenty of space is still available. For example, a 6 TB

volume that is 97% used still has about 200 GB available. In this case, you can define the thresholds as available space in megabytes (MB) instead of used space percentage, with these environment variables:

DISKCHECKALERT_FREE_MB

A warning is created when the available space falls below this value. When set, it replaces the DISKCHECKALERT percentage threshold.

DISKTHRESHOLD_FREE_MB

A critical alert is created, and the container health check fails, when the available space falls below this value. When set, it replaces the DISKTHRESHOLD percentage threshold.

These variables are not set by default, so the percentage thresholds are used. Each variable only replaces the percentage threshold of its own level. For example, if you only set DISKTHRESHOLD_FREE_MB, warnings are still created using DISKCHECKALERT. The values must be non-negative integers.

These variables can only be set as environment variables. They have no equivalent in /etc/rhn/rhn.conf.

Procedure: Configuring Free Space Thresholds

1. Create systemd override directories:

```
mkdir -p /etc/systemd/system/uyuni-server.service.d/
mkdir -p /etc/systemd/system/uyuni-db.service.d/
```

2. Create configuration files for both server and database services. This example creates a warning below 200 GB and a critical alert below 50 GB of available space:

```
cat > /etc/systemd/system/uyuni-server.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT_FREE_MB=204800 --env
DISKTHRESHOLD_FREE_MB=51200"
EOF

cat > /etc/systemd/system/uyuni-db.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT_FREE_MB=204800 --env
DISKTHRESHOLD_FREE_MB=51200"
EOF
```



Each override file can only contain one PODMAN_EXTRA_ARGS value. To set several disk check variables, such as DISKCHECKDIRS and the thresholds, add all the --env options to the same line.

3. Apply the configuration by restarting Uyuni:

```
mgradm restart
```



The same thresholds apply to every monitored directory. If the monitored directories are on file systems of very different sizes, choose values that leave enough headroom on the smallest one.

11.3. Database Disk Check

The database container has its own disk check that monitors only the PostgreSQL data directory (/var/lib/pgsql/data). This directory is always monitored and cannot be changed or overridden with the DISKCHECKDIRS environment variable.

The database disk check uses the DISKCHECKALERT, DISKTHRESHOLD, DISKCHECKALERT_FREE_MB, and DISKTHRESHOLD_FREE_MB environment variables configured for the uyuni-db service.

The database disk check runs during the container health checks and when the container starts. When the critical threshold is exceeded, the health check will fail and the container will be automatically stopped to prevent data corruption. The container does not start while the critical threshold is exceeded.

The warnings and notifications for the database are created by a PostgreSQL function that reads the thresholds every time it runs. Changes to the thresholds take effect after you restart Uyuni.



The database container only monitors /var/lib/pgsql/data. You cannot configure additional directories to monitor in the database container using DISKCHECKDIRS. To monitor other directories, use the server container configuration.

11.4. Disable Space Checking

The space checking tool is enabled by default and runs hourly via a Taskomatic scheduled task. Notification can be disabled by disabling diskcheck-task-queue-default Taskomatic job. To effectively disable disk space monitoring, set the threshold values to 99 (percent). If you use the free space thresholds, set DISKCHECKALERT_FREE_MB and DISKTHRESHOLD_FREE_MB to 0 instead, because they replace the percentage thresholds.

Procedure: Disabling Disk Space Monitoring

1. Edit the systemd override files to set very high thresholds:

```
cat > /etc/systemd/system/uyuni-server.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT=99 --env DISKTHRESHOLD=99"
EOF

cat > /etc/systemd/system/uyuni-db.service.d/diskcheck.conf << 'EOF'
[Service]
Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKALERT=99 --env DISKTHRESHOLD=99"
EOF
```

2. Apply the configuration by restarting Uyuni:

```
mgradm restart
```



Disabling disk space monitoring is not recommended in production environments, as running out of disk space can cause severe and potentially unrecoverable damage to the database.

Chapter 12. Image Building and Management

12.1. Image building overview

Uyuni enables system administrators to build containers and OS Images and push the results to an image store.

Procedure: Building and pushing images

1. Define an image store.
2. Define an image profile and associate it with a source (either a git repository or a directory).
3. Build the image.
4. Push the image to the image store.

Uyuni supports these two build types: Dockerfile and the Kiwi build type. The Kiwi build type is used to build system, virtual, and other images.

The image store for the Kiwi build type is pre-defined as a file system directory in the srv-www volume.

The image files can be downloaded from https://UYUNI-HOST/os-images/ORGANIZATION-ID/FILE-NAME. The exact location can be determined from the image details page.

12.2. Container images

12.2.1. Requirements

The containers feature is available for Salt clients running SUSE Linux Enterprise Server 12 or later. Before you begin, ensure your environment meets these requirements:

- A published git repository containing a Dockerfile and configuration scripts. The repository can be public or private, and should be hosted on GitHub, GitLab, or BitBucket.
- A properly configured image store, such as a Docker registry.

For more information on containers, see https://documentation.suse.com/container/all/html/Container-guide/.

12.2.2. Create a build host

To build images with Uyuni, you need to create and configure a build host. Container build hosts are Salt clients

running SUSE Linux Enterprise 12 or later. This section guides you through the initial configuration for a build host.



The operating system on the build host must match the operating system on the targeted image.

For example, build SUSE Linux Enterprise Server 15 based images on a build host running SUSE Linux Enterprise Server 15 (SP2 or later) OS version. Build SUSE Linux Enterprise Server 12 based images on a build host running SUSE Linux Enterprise Server 12 SP5 or SUSE Linux Enterprise Server 12 SP4 OS version.

Cross-architecture builds are not supported.

From the Uyuni Web UI, perform these steps to configure a build host:

Procedure: Building a host

1. Select a Salt client to be designated as a build host from the **Systems › Systems** overview page.
2. From the System Details page of the selected client assign the containers modules. Navigate to **Software › Software Channels** and enable the containers module (for example, SLE-Module-Containers15-Pool and SLE-Module-Containers15-Updates). Continue with **[Next]**.
3. Schedule the ` Software Channel Change`, and click **[Confirm]**.
4. From the System Details tab select **Properties** page, and enable Container Build Host from the Add-on System Types list. Confirm by clicking **[Update Properties]**.
5. Install all required packages by applying Highstate. From the system details tab select **States › Highstate**, and click **[Apply Highstate]**. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

12.2.3. Create an activation key for containers

The containers built with Uyuni use channels associated to the activation key as repositories when building the image. This section guides you through creating an ad-hoc activation key for this purpose.

To build a container, you need an activation key that is associated with a channel other than SUSE Manager Default.

Procedure: Creating an Activation Key

1. In the Uyuni Web UI, as an administrator, navigate to **Systems › Activation Keys**.
2. Click the **[Create Key]** button.
3. On the Activation Key Details page, in the Description field, enter a description of the activation key.
4. In the Key field, enter a name for the activation key. For example, SLES15-SP7 for SUSE Linux Enterprise Server 15 SP7.

   - Do not use commas or double quotes in the Key field for any SUSE products. However, you **must** use commas for Red Hat Products.
   - All other characters are allowed, but <> (){} (this includes the space) are removed automatically.
   - If the field is left empty, a random string is generated.

5. In the Usage add number of times the key can be used, or leave it blank for unlimited usage.
6. In the Base Channels drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate. For more information, see **Reference Guide › Admin › Setup Wizard** and **Administration Guide › Custom Channels**.
7. Select the Child Channels you need (for example, the mandatory SUSE Multi-Linux Manager tools and updates channels).
8. Check the Add-On System Types checkbox if you need to enable any of the options. For more information about system types, see **Client Configuration Guide › System Types**.

9. We recommend you leave the Contact Method set to Default. For more information about contact methods, see **Client Configuration Guide › Contact Methods Intro**.
10. We recommend you leave the Universal Default setting unchecked.
11. Click **[Create Activation Key]** to create the activation key.
12. After the activation key is created, the page shows additional checkbox Configuration File Deployment below the list of Add-On System Types. To enable deploying configuration files to systems on registration, select this checkbox.
13. Click **[Update Activation Key]** to save the change.

For more information, see **Client Configuration Guide › Activation Keys**.

12.2.4. Create an image store

All built images are pushed to an image store. This section contains information about creating an image store. Image store is generally referenced as a registry.

Procedure: Creating image store

1. Select **Images › Stores**.
2. Click Create to create a new store.
3. From the Store Type select the correct type.
4. Define a name for the image store in the Label field.
5. Provide the path to your image registry by filling in the URI field, as a fully qualified domain name (FQDN) for the container registry host (whether internal or external).

   ```
   registry.example.com
   ```

   The Registry URI can also be used to specify an image store on a registry that is already in use.

```
registry.example.com:5000/myregistry/myproject
```

6. Click [Create] to add the new image store.

12.2.5. Create an image profile

All container images are built using an image profile, which contains the building instructions. This section contains information about creating an image profile with the Uyuni Web UI.

Procedure: Creating an image profile

1. To create an image profile select **Images › Profiles** and click [Create].
2. Provide a name for the image profile by filling in the Label field.

> If your container image tag is in a format such as myproject/myimage, make sure your image store registry URI contains the /myproject suffix.

3. Use Dockerfile as the Image Type.
4. Use the drop-down menu to select your registry from the Target Image Store field.
5. In the Path field, type a GitHub, GitLab, or BitBucket repository URL. The path can also be a local directory on the build host. The URL should be http, https, or a token authentication URL. For GitHub or GitLab, use one of these formats:

GitHub Path Options

- GitHub single user project repository

```
https://github.com/USER/project.git#branchname:folder
```

- GitHub organization project repository

```
https://github.com/ORG/project.git#branchname:folder
```

- GitHub token authentication

  If your git repository is private, modify the profile's URL to include authentication. Use this URL format to authenticate with a GitHub token:

```
https://USER:<AUTHENTICATION_TOKEN>@github.com/USER/project.git#master:/con
tainer/
```

- GitLab single user project repository

```
https://gitlab.example.com/USER/project.git#master:/container/
```

- GitLab groups project repository

```
https://gitlab.example.com/GROUP/project.git#master:/container/
```

- GitLab token authentication

  If your git repository is private and not publicly accessible, you need to modify the profile's git URL to include authentication. Use this URL format to authenticate with a GitLab token:

```
https://gitlab-ci-
token:<AUTHENTICATION_TOKEN>@gitlab.example.com/USER/project.git#master:/co
ntainer/
```



If you do not specify a git branch, the master branch is used by default. If a folder is not specified, the image sources (Dockerfile sources) are expected to be in the root directory of the GitHub or GitLab checkout.

6. Select an Activation Key. Activation keys ensure that images using a profile are assigned to the correct channel and packages.



When you associate an activation key with an image profile you are ensuring any image using the profile uses the correct software channel and any packages in the channel.

7. Click the [Create] button.

12.2.5.1. Example Dockerfile sources

An Image Profile that can be reused is published at https://github.com/SUSE/manager-build-profiles.



The ARG parameters ensure that the built image is associated with the desired repository served by Uyuni. The ARG parameters also allow you to build image versions of SUSE Linux Enterprise Server which may differ from the version of SUSE Linux Enterprise Server used by the build host itself.

For example: The ARG repo parameter and the echo command pointing to the repository file, creates and then injects the correct path into the repository file for the desired channel version.

The repository is determined by the activation key that you assigned to your image profile.

```
FROM registry.example.com/sles12sp2
MAINTAINER Tux Administrator "tux@example.com"

### Begin: These lines are required for use with {productname}

ARG repo
ARG cert

# Add the correct certificate
RUN echo "$cert" > /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT.pem

# Update certificate trust store
RUN update-ca-certificates

# Add the repository path to the image
RUN echo "$repo" > /etc/zypp/repos.d/susemanager:dockerbuild.repo

### End: These lines are required for use with {productname}

# Add the package script
ADD add_packages.sh /root/add_packages.sh

# Run the package script
RUN /root/add_packages.sh

# After building remove the repository path from image
RUN rm -f /etc/zypp/repos.d/susemanager:dockerbuild.repo
```

12.2.5.2. Using custom info key-value pairs as Docker buildargs

You can assign custom info key-value pairs to attach information to the image profiles. Additionally, these key-value pairs are passed to the Docker build command as buildargs.

For more information about the available custom info keys and creating additional ones, see **Reference Guide ›**

Systems › Custom System Info.

12.2.6. Build an image

There are two ways to build an image. The first way is to create it from scratch. To do that, select **Images › Build** from the left navigation bar, or click the build icon in the **Images › Profiles** list and follow the procedure.

Procedure: Building an image

1. Select **Images › Build**.
2. Add a different tag name if you want a version other than the default latest (only relevant to containers).
3. Select Build Profile and Build Host.



Notice the Profile Summary to the right of the build fields. When you have selected a build profile, detailed information about the selected profile is displayed in this area.

4. To schedule a build click the **[Build]** button.

12.2.7. Import an image

The second way to get an image is to import and inspect arbitrary images. To do that, select **Images › Image List** from the left navigation bar. Complete the text boxes of the Import dialog. When it has processed, the imported image is listed on the Image List page.

Procedure: Importing an image

1. From **Images › Image list** click **[Import]** to open the Import Image dialog.
2. In the Import Image dialog complete these fields:

Image store

The registry from where the image is pulled for inspection.

Image name

The name of the image in the registry.

Image version

The version of the image in the registry.

Build host

The build host that pulls and inspects the image.

Activation key

The activation key that provides the path to the software channel that the image is inspected with.

3. For confirmation, click **[Import]**.

The entry for the image is created in the database, and an Inspect Image action on Uyuni is scheduled.

When it has been processed, you can find the imported image in the Image List. It has a different icon in the Build column, to indicate that the image is imported. The status icon for the imported image can also be seen on the Overview tab for the image.

12.2.8. Troubleshooting

12.2.8.1. Image Inspection

Base container images (BCIs) come with all the software to run them, but because they are lightweight, they may not include all the tools and libraries required for inspection.

When performing container image inspection, Uyuni copies and extracts the Salt bundle (venv-salt-minion) from the build host (where the inspection is scheduled) into the target container, and then executes it inside the container. Therefore, the target container image must satisfy any dynamic dependencies of the build host's Salt bundle.

If the target image is missing required dynamic libraries, the inspection fails. For example, you might see error messages like:

```
libssl.so.1.1: cannot open shared object file: No such file or directory
```

The most common dependency issue is with OpenSSL libraries, because the Salt bundle's Python modules are

dynamically linked against the build host's OpenSSL version. Because OpenSSL versions differ between major operating system versions (such as SLE 15 and SLE 16), compatibility depends on both the build host and the target image:

- **On a SLE 15 build host**: The Salt bundle is compiled against OpenSSL 1.1 (providing libssl.so.1.1). Therefore, any image inspected on a SLE 15 build host must have OpenSSL 1.1 installed (typically via the libopenssl1_1 package). Since OpenSSL 1.1 is not available on SLE 16 and openSUSE Leap 16, a SLE 16 / Leap 16 image **cannot** be inspected on a SLE 15 build host.
- **On a SLE 16 build host**: The Salt bundle is compiled against OpenSSL 3 (providing libssl.so.3). Therefore, any image inspected on a SLE 16 build host must have OpenSSL 3 installed (typically via the libopenssl3 package).

To ensure successful image inspection, follow these best practices:

- Match the major operating system version of the build host and the target image. For example, use a SLE 16 / Leap 16 build host to inspect SLE 16 / Leap 16 images, and a SLE 15 build host to inspect SLE 15 images.
- Install the correct version of OpenSSL inside your image (for example, by adding libopenssl3 for SLE 16 or libopenssl1_1 for SLE 15 to the Dockerfile) and rebuilding the image.

The same dependency requirements apply to other dynamic libraries, such as libexpat.

12.2.8.2. General issues

These are some known problems when working with images:

- HTTPS certificates to access the registry or the git repositories should be deployed to the client by a custom state file.
- SSH git access using Docker is currently unsupported.

12.3. OS images

OS Images are built by the Kiwi build system. The output image is customizable and can be PXE, QCOW2, LiveCD, or other types of images.

For more information about the Kiwi build system, see the Kiwi documentation.

12.3.1. Requirements

The Kiwi image building feature is available for Salt clients running SUSE Linux Enterprise Server 12 and SUSE Linux Enterprise Server 11.

Kiwi image configuration files and configuration scripts must be accessible in one of these locations:

- Git repository
- HTTP or HTTPS hosted tar archive
- Local directory on the build host

For an example of a complete Kiwi repository served by git, see https://github.com/SUSE/manager-build-profiles/tree/master/OSImage.



You need at least 1 GB of RAM available for hosts running OS Images built with Kiwi. Disk space depends on the actual size of the image. For more information, see the documentation of the underlying system.

12.3.2. Accessing Git repositories via an HTTP/HTTPS proxy when building images

When a build host needs to fetch sources from a git repository that is only reachable through an HTTP/HTTPS proxy, you may see git timeouts during the build because the git client invoked by the Salt state does not always pick up system-wide proxy settings (for example /etc/sysconfig/proxy or environment variables).

To make git use the proxy for non-interactive builds create /etc/gitconfig with an http.proxy entry:

```
# /etc/gitconfig
[http]
    proxy = http://proxy.example.com:3128
# or for HTTPS
[http]
    proxy = https://proxy.example.com:3128
```

If the proxy requires authentication, prefer a credential helper or use a credential store instead of embedding credentials in the URL.

12.3.2.1. Automating with Salt

You can manage /etc/gitconfig on build hosts through Configuration Management, the same way as on any other managed system. Create a Salt state file and assign it to the build host via a config channel, then apply a highstate from the Uyuni UI.

If you want to source the proxy address from a System Custom Info field, use the custom_info: pillar prefix:

```
/etc/gitconfig:
  file.managed:
    - user: root
    - group: root
    - mode: '0644'
```

```
    - contents: |
        [http]
            proxy = {{ salt['pillar.get']('custom_info:build_server:git_proxy',
'http://proxy.example.com:3128') }}
```

12.3.3. Container-based Kiwi image build support

Uyuni introduces a container-based Kiwi image build system, in addition to the existing legacy Kiwi and KiwiNG tools.

12.3.3.1. Configuration and overrides

Administrators can override the default behavior using the following pillar or custom values. To configure these, navigate to menu:[Systems>Custom System Info] in the Web UI and create the necessary keys.

The build system used depends on the underlying OS or specific pillar values:

- for SLE 11 / SLE 12: legacy Kiwi v7
- for SLE 15: KiwiNG (v9 and containerized Kiwi 10)

Administrators can override the default behavior using the following pillar or custom values:

- use_kiwi_ng: force the use of Kiwi 9,
- use_kiwi_container: force the use of containerized Kiwi 10. To enable this, set the value to 1.
- use_bundle_build: upload additional KIWI bundle build artifacts (like .install.tar which contains PXE files, and .raw.xz) to the server. To enable this, set the value to true.

12.3.3.2. Version-specific configurations

When using a containerized build host for SUSE Linux Enterprise 15 profiles, specific configurations are required because SLES 15 profiles rely on Kiwi 9, while the default container behavior uses Kiwi 10.

To ensure the correct version is used for SLES 15 profiles, you must define the kiwi_image custom info key with the following value:

- Key: kiwi_image
- Value: registry.suse.com/bci/kiwi:9

If this key is not set, the system defaults to the latest version (e.g., registry.suse.com/bci/kiwi:10.2), which may result in build issues for SLES 15 profiles.

12.3.4. Create a build host

To build all kinds of images with Uyuni, create and configure a build host. OS Image build hosts are Salt clients running on SUSE Linux Enterprise Server 15 (SP2 or later) or SUSE Linux Enterprise Server 12 (SP4 or later).

This procedure guides you through the initial configuration for a build host.



The operating system on the build host must match the operating system on the targeted image.

For example, build SUSE Linux Enterprise Server 15 based images on a build host running SUSE Linux Enterprise Server 15 (SP2 or later) OS version. Build SUSE Linux Enterprise Server 12 based images on a build host running SUSE Linux Enterprise Server 12 SP5 or SUSE Linux Enterprise Server 12 SP4 OS version.

Cross-architecture builds are not possible. For example, you must build Raspberry PI SUSE Linux Enterprise Server 15 SP3 image on a Raspberry PI (aarch64 architecture) build host running SUSE Linux Enterprise Server 15 SP3.

Procedure: Configure the build host in the Uyuni Web UI

1. Select a client to be designated as a build host from the **Systems › Overview** page.
2. Navigate to the **System Details › Properties** tab, and check the Add-on System Type > OS Image Build Host box.
3. Confirm with **[Update Properties]**.
4. Navigate to **System Details › Software › Software Channels**, and enable the required software channels depending on the build host version.
    - SUSE Linux Enterprise Server 12 build hosts require Uyuni Client tools (SLE-Manager-Tools12-Pool and SLE-Manager-Tools12-Updates).
    - SUSE Linux Enterprise Server 15 build hosts require SUSE Linux Enterprise Server modules SLE-Module-DevTools15-SP4-Pool and SLE-Module-DevTools15-SP4-Updates.
    - Schedule and click **[Confirm]**.
5. Install Kiwi and all required packages by applying Highstate. From the

system details page select **States › Highstate** and click **[Apply Highstate]**. Alternatively, apply Highstate from the Uyuni Server command line:

```
salt '$your_client' state.highstate
```

12.3.4.1. Uyuni web server public certificate RPM

Build host provisioning copies the Uyuni certificate RPM to the build host. This certificate is used for accessing repositories provided by Uyuni.

The certificate is packaged in RPM by the mgr-package-rpm-certificate-osimage package script. The package script is called automatically during a new Uyuni installation.

When you upgrade the spacewalk-certs-tools package, the upgrade scenario calls the package script using the default values. However if the certificate path was changed or unavailable, call the package script manually using --ca-cert-full-path <path_to_certificate> after the upgrade procedure has finished.

12.3.4.2. Package script call example

```
/usr/sbin/mgr-package-rpm-certificate-osimage --ca-cert-full-path /root/ssl-build/RHN-ORG-TRUSTED-SSL-CERT
```

The RPM package with the certificate is stored in a salt-accessible directory such as:

```
/usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm
```

The RPM package with the certificate is provided in the local build host repository:

```
/var/lib/Kiwi/repo
```


Specify the RPM package with the Uyuni SSL certificate in the build source, and make sure your Kiwi configuration contains rhn-org-trusted-ssl-cert-osimage as a required package in the bootstrap section.

Listing 3. config.xml

```
...
  <packages type="bootstrap">
    ...
    <package name="rhn-org-trusted-ssl-cert-osimage" bootinclude="true"/>
  </packages>
...
```

12.3.5. Create an activation key for OS images

Create an activation key associated with the channel that your OS Images can use as repositories when building the image.

Activation keys are mandatory for OS Image building.



To build OS Images, you need an activation key that is associated with a channel other than Default activation key.

Procedure: Creating an Activation Key

1. In the Uyuni Web UI, as an administrator, navigate to **Systems › Activation Keys**.
2. Click the **[Create Key]** button.
3. On the Activation Key Details page, in the Description field, enter a description of the activation key.
4. In the Key field, enter a name for the activation key. For example, SLES15-SP7 for SUSE Linux Enterprise Server 15 SP7.



- Do not use commas or double quotes in the Key field for any SUSE products. However, you **must** use commas for Red Hat Products.
- All other characters are allowed, but <> (){} (this includes the space) are removed automatically.
- If the field is left empty, a random string is generated.

5. In the Usage add number of times the key can be used, or leave it blank for unlimited usage.
6. In the Base Channels drop-down box, select the appropriate base software channel, and allow the relevant child channels to populate. For more information, see **Reference Guide › Admin › Setup Wizard** and **Administration Guide › Custom Channels**.

7. Select the Child Channels you need (for example, the mandatory SUSE Multi-Linux Manager tools and updates channels).
8. Check the Add-On System Types checkbox if you need to enable any of the options. For more information about system types, see **Client Configuration Guide › System Types**.
9. We recommend you leave the Contact Method set to Default. For more information about contact methods, see **Client Configuration Guide › Contact Methods Intro**.
10. We recommend you leave the Universal Default setting unchecked.
11. Click **[Create Activation Key]** to create the activation key.
12. After the activation key is created, the page shows additional checkbox Configuration File Deployment below the list of Add-On System Types. To enable deploying configuration files to systems on registration, select this checkbox.
13. Click **[Update Activation Key]** to save the change.

For more information, see **Client Configuration Guide › Activation Keys**.

12.3.6. Create an image store

OS Images can require a significant amount of storage space. By default, the image store is using the srv-www volume.



Image stores for Kiwi build type, used to build system, virtual, and other images, are not supported yet.

The image files can be downloaded from https://UYUNI-HOST/os-images/ORGANIZATION-ID/FILE-NAME. The exact location can be determined from the image details page.

12.3.7. Create an image profile

Manage image profiles using the Web UI.

Procedure: Creating an image profile

1. To create an image profile select from **Images › Profiles** and click **[Create]**.
2. In the Label field, provide a name for the Image Profile.
3. Use Kiwi as the Image Type.
4. Image store is automatically selected.
5. Enter a Config URL to the directory containing the Kiwi configuration files. For example, a git URI such as https://github.com/SUSE/manager-build-profiles#master:OSImage/SLE-Micro54. Other options are a HTTP or HTTPS hosted tar archive or a local directory on the build host. For more information, see source format options at the end of this section.
6. Enter Kiwi options if needed. If the Kiwi configuration files specify multiple profiles, use --profile <name> to select the active one. For other options, see Kiwi documentation.
7. Select an Activation Key. Activation keys ensure that images using a profile are assigned to the correct channel and packages.

   Associate an activation key with an image profile to ensure the image profile uses the correct software channel, and any packages.

8. Confirm with the **[Create]** button.

Source Format Options

- git/HTTP(S) URL to the repository

  URL to a public or private git repository containing the sources of the image to be built. Depending on the layout of the repository the URL can be:

  ```
  https://github.com/SUSE/manager-build-profiles
  ```

  You can specify a branch after the # character in the URL. In this example,

we use the master branch:

```
https://github.com/SUSE/manager-build-profiles#master
```

You can specify a directory that contains the image sources after the : character. In this example, we use OSImage/POS_Image-JeOS6:

```
https://github.com/SUSE/manager-build-profiles#master:OSImage/POS_Image-JeOS6
```

- HTTP(S) URL to the tar archive

  URL to the tar archive, compressed or uncompressed, hosted on the webserver.

  ```
  https://myimagesourceserver.example.org/MyKiwiImage.tar.gz
  ```

- Path to the directory on the build host

  Enter the path to the directory with the Kiwi build system sources. This directory must be present on the selected build host.

  ```
  /var/lib/Kiwi/MyKiwiImage
  ```

12.3.7.1. Example of Kiwi sources

Kiwi sources consist at least of config.xml. Usually, config.sh and images.sh are present as well. Sources can also contain files to be installed in the final image under the root subdirectory.

For information about the Kiwi build system, see the Kiwi documentation.

SUSE provides examples of fully functional image sources at the SUSE/manager-build-profiles public GitHub repository.

Listing 4. Example of JeOS config.xml

```
<?xml version="1.0" encoding="utf-8"?>

<image schemaversion="6.1" name="POS_Image_JeOS6">
    <description type="system">
        <author>Admin User</author>
        <contact>noemail@example.com</contact>
        <specification>SUSE Linux Enterprise 12 SP3 JeOS</specification>
```

```
    </description>
    <preferences>
        <version>6.0.0</version>
        <packagemanager>zypper</packagemanager>
        <bootsplash-theme>SLE</bootsplash-theme>
        <bootloader-theme>SLE</bootloader-theme>

        <locale>en_US</locale>
        <keytable>us.map.gz</keytable>
        <timezone>Europe/Berlin</timezone>
        <hwclock>utc</hwclock>

        <rpm-excludedocs>true</rpm-excludedocs>
        <type boot="saltboot/suse-SLES12" bootloader="grub2" checkprebuilt="true"
compressed="false" filesystem="ext3" fsmountoptions="acl" fsnocheck="true" image="pxe"
kernelcmdline="quiet"></type>
    </preferences>
    <!--    CUSTOM REPOSITORY
    <repository type="rpm-dir">
      <source path="this://repo"/>
    </repository>
    -->
    <packages type="image">
        <package name="patterns-sles-Minimal"/>
        <package name="aaa_base-extras"/> <!-- wouldn't be SUSE without that ;-) -->
        <package name="kernel-default"/>
        <package name="venv-salt-minion"/>
        ...
    </packages>
    <packages type="bootstrap">
        ...
        <package name="sles-release"/>
        <!-- this certificate package is required to access {productname} repositories
             and is provided by {productname} automatically -->
        <package name="rhn-org-trusted-ssl-cert-osimage" bootinclude="true"/>

    </packages>
    <packages type="delete">
        <package name="mtools"/>
        <package name="initviocons"/>
        ...
    </packages>
</image>
```

12.3.8. Build an image

There are two ways to build or get an image using the Web UI. Either select **Images › Build**, or click the build icon in the **Images › Profiles** list.

Procedure: Building an image

1. Select **Images › Build**.
2. Add a different tag name if you want a version other than the default latest (applies only to containers).

3. Select the Image Profile and a Build Host.



A Profile Summary is displayed to the right of the build fields. When you have selected a build profile, detailed information about the selected profile is shown here.

4. To schedule a build, click the [Build] button.



The build server cannot run any form of automounter during the image building process. If applicable, ensure that you do not have your Gnome session running as root. If an automounter is running, the image build finishes successfully, but the checksum of the image is different and causes a failure.

After the image is successfully built, the inspection phase begins. During the inspection phase SUSE Multi-Linux Manager collects information about the image:

- List of packages installed in the image
- Checksum of the image
- Image type and other image details



If the built image type is PXE, a Salt pillar is also generated. Image pillars are stored in the database and the Salt subsystem can access details about the generated image. Details include where the image files are located and provided, image checksums, information needed for network boot, and more.

The generated pillar is available to all connected clients.

12.3.9. Troubleshooting

Building an image requires several dependent steps. When the build fails, investigating Salt states results and build log can help identify the source of the failure. You can carry out these checks when the build fails:

- The build host can access the build sources
- There is enough disk space for the image on both the build host and the Uyuni server
- The activation key has the correct channels associated with it
- The build sources used are valid
- The RPM package with the Uyuni public certificate is up to date and available at /usr/share/susemanager/salt/images/rhn-org-trusted-ssl-cert-osimage-1.0-1.noarch.rpm. For more on how to refresh a public certificate RPM, see Create a build host.

12.3.10. Limitations

The section contains some known issues when working with images.

- HTTPS certificates used to access the HTTP sources or git repositories should be deployed to the client by a custom state file, or configured manually.
- Importing Kiwi-based images is not supported.

12.4. List of built images

To list available built images select **Images › Image List**. A list of all images is displayed.

Displayed data about images includes an image Name, its Version, Revision, and the build Status. You can also see the image update status with a listing of possible patch and package updates that are available for the image.

For OS Images, the Name and Version fields originate from Kiwi sources and are updated at the end of successful build. During building or after failed build these fields show a temporary name based on profile name.

Revision is automatically increased after each successful build. For OS Images, multiple revisions can co-exist in the store.

For Container Images the store holds only the latest revision. Information about previous revisions (packages, patches, etc.) are preserved and it is possible to list them with the Show obsolete checkbox.

Clicking the **[Details]** button on an image provides a detailed view. The detailed view includes an exact list of relevant patches, list of all packages installed within the image and a build log.

Clicking the **[Delete]** button deletes the image from the list. It also deletes the associated pillar, files from OS Image Store and obsolete revisions.



The patch and the package list is only available if the inspect state after a build was successful.

Chapter 13. Infrastructure maintenance tasks

If you work with scheduled downtime periods, you might find it difficult to remember all the things that you need to do before, during, and after that critical downtime of the Uyuni Server. Uyuni Server related systems such as Inter-Server Synchronization Slave Servers or Uyuni Proxies are also affected and have to be considered.

SUSE recommends you always keep your Uyuni infrastructure updated. That includes servers, proxies, and build hosts. If you do not keep the Uyuni Server updated, you might not be able to update some parts of your environment when you need to.

This section contains a checklist for your downtime period, with links to further information on performing each of the steps.

13.1. Server

Procedure: Server checks

1. Apply the latest updates.
2. Upgrade to the latest service pack, if required.
3. Run `podman ps` and check whether all required services are up and running.

On SUSE Linux Enterprise Server 15 SP7, you can install updates using a package manager:

- For information on using YaST, see https://documentation.suse.com/sles/15-SP7/html/SLES-all/cha-onlineupdate-you.html.
- For information on using zypper, see https://documentation.suse.com/sles/15-SP7/html/SLES-all/cha-sw-cl.html#sec-zypper.

On SL Micro 6.2, you can install updates using the `transactional-update` command. For information on using `transactional-update`, see https://documentation.suse.com/sle-micro/6.2/html/Micro-transactional-updates/index.html.

By default, several update channels are configured and enabled for the Uyuni Server. New and updated packages become available automatically.

13.1.1. Client tools

When the server is updated consider updating some tools on the clients, too. Updating `venv-salt-minion`,

zypper, and other related management package on clients is not a strict requirement, but it is a best practice in general. For example, a maintenance update on the server might introduce a major new Salt version. Then Salt clients continue to work but might experience problems later on. To avoid this SUSE makes sure that venv-salt-minion will always be updated safely.

13.2. Inter-Server Synchronization slave server

If you are using an Inter-Server Synchronization slave server, update it after the Uyuni Server update is complete.

For more information, see **Specialized Guides › Large Deployments › Inter-Server Synchronization - Version 2**.

13.3. Monitoring server

If you are using a monitoring server for Prometheus, update it after the Uyuni Server update is complete.

For more information on monitoring, see **Administration Guide › Monitoring**.

13.4. Proxy

Proxies should be updated as soon as Uyuni Server updates are complete.

In general, running a proxy connected to a server on a different version is not supported. The only exception is for the duration of updates where it is expected that the server is updated first, so the proxy could run the previous version temporarily.



Always upgrade the server first, then any proxy.

Chapter 14. Live Patching with Uyuni

Performing a kernel update usually requires a system reboot. Common vulnerability and exposure (CVE) patches should be applied as soon as possible, but if you cannot afford the downtime, you can use Live Patching to inject these important updates and skip the need to reboot.

The procedure for setting up Live Patching is slightly different for SLES 12 and SLES 15. Both procedures are documented in this section.

14.1. Set up Channels for Live Patching

A reboot is required every time you update the full kernel package. Therefore, it is important that clients using Live Patching do not have newer kernels available in the channels they are assigned to. Clients using live patching have updates for the running kernel in the live patching channels.

There are two ways to manage channels for live patching:

Use content lifecycle management to clone the product tree and remove kernel versions newer than the running one. This procedure is explained in the **Administration Guide › Content Lifecycle Examples › Enhance Project with Livepatching**. This is the recommended solution.

Alternatively, use the spacewalk-manage-channel-lifecycle tool. This procedure is more manual and requires command line tools as well as the Web UI. This procedure is explained in this section for SLES 15 SP5, but it also works for SLE 12 SP4 or later.

14.1.1. Use spacewalk-manage-channel-lifecycle for Live Patching



spacewalk-manage-channel-lifecycle has been deprecated and will be removed in an upcoming release. Users are advised to switch to the supported and feature-rich Content Lifecycle Management (CLM) API instead.

Cloned vendor channels should be prefixed by dev for development, testing, or prod for production. In this procedure, you create a dev cloned channel and then promote the channel to testing.

Procedure: Cloning Live Patching Channels

1. At the command prompt on the client, as root, obtain the current package channel tree:

```
# spacewalk-manage-channel-lifecycle --list-channels
Spacewalk Username: admin
Spacewalk Password:
Channel tree:

 1. sles15-sp7-pool-x86_64
```

```
      \__ sle-live-patching15-pool-x86_64-sp7
      \__ sle-live-patching15-updates-x86_64-sp7
      \__ sle-manager-tools15-pool-x86_64-sp7
      \__ sle-manager-tools15-updates-x86_64-sp7
      \__ sles15-sp7-updates-x86_64
```

2. Use the spacewalk-manage-channel command with the init argument to automatically create a new development clone of the original vendor channel:

```
spacewalk-manage-channel-lifecycle --init -c sles15-sp7-pool-x86_64
```

3. Check that dev-sles15-sp7-updates-x86_64 is available in your channel list.

Check the dev cloned channel you created, and remove any kernel updates that require a reboot.

Procedure: Removing Non-Live Kernel Patches from Cloned Channels

1. Check the current kernel version by selecting the client from **Systems › System List**, and taking note of the version displayed in the Kernel field.
2. In the Uyuni Web UI, select the client from **Systems › Overview**, navigate to the **Software › Manage › Channels** tab, and select dev-sles15-sp7-updates-x86_64. Navigate to the Patches tab, and click **[List/Remove Patches]**.
3. In the search bar, type kernel and identify the kernel version that matches the kernel currently used by your client.
4. Remove all kernel versions that are newer than the currently installed kernel.

Your channel is now set up for live patching, and can be promoted to testing. In this procedure, you also add the live patching child channels to your client, ready to be applied.

Procedure: Promoting Live Patching Channels

1. At the command prompt on the client, as root, promote and clone the dev-sles15-sp7-pool-x86_64 channel to a new testing channel:

```
# spacewalk-manage-channel-lifecycle --promote -c dev-sles15-sp7-pool-x86_64
```

2. In the Uyuni Web UI, select the client from **Systems › Overview**, and navigate to the **Software › Software Channels** tab.
3. Check the new test-sles15-sp7-pool-x86_64 custom channel to change the base channel, and check both corresponding live patching child channels.
4. Click **[Next]**, confirm that the details are correct, and click **[Confirm]** to save the changes.

You can now select and view available CVE patches, and apply these important kernel updates with Live

Patching.

14.2. Live Patching on SLES 15

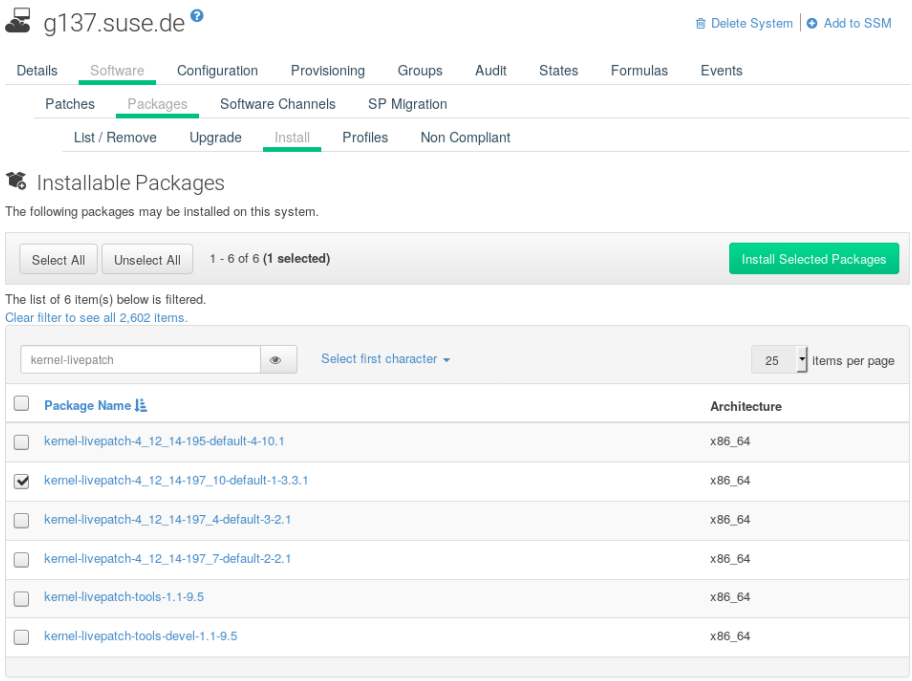
On SLES 15 systems and newer, live patching is managed by the klp livepatch tool.

Before you begin, ensure:

- Uyuni is fully updated.
- You have one or more Salt clients running SLES 15 (SP1 or later).
- Your SLES 15 Salt clients are registered with Uyuni.
- You have access to the SLES 15 channels appropriate for your architecture, including the live patching child channel (or channels).
- The clients are fully synchronized.
- Assign the clients to the cloned channels prepared for live patching. For more information on preparation, see **Administration Guide › Live Patching Channel Setup**.

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from **Systems › Overview**, and navigate to the **Software › Packages › Install** tab. Search for the kernel-livepatch package, and install it.



2. Apply the highstate to enable Live Patching, and reboot the client.

3. Repeat for each client that you want to manage with Live Patching.
4. To check that live patching has been enabled correctly, select the client from **Systems › System List**, and ensure that Live Patch appears in the Kernel field.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni Web UI, select the client from **Systems › Overview**. A banner at the top of the screen shows the number of critical and non-critical packages available for the client.
2. Click **[Critical]** to see a list of the available critical patches.
3. Select any patch with a synopsis reading Important: Security update for the Linux kernel. Security bugs also include their CVE number, where applicable.
4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in **Audit › CVE Audit**, and apply the patch to any clients that require it.



- Not all kernel patches are Live Patches. Non-Live kernel patches are represented by a Reboot Required icon located next to the Security shield icon. These patches always require a reboot.
- Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and requires a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit displays this requirement.

14.3. Live Patching on SLES 12

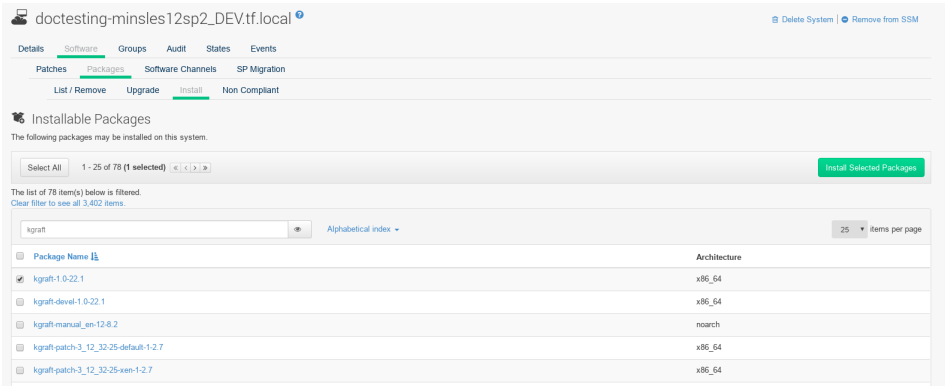
On SLES 12 systems, live patching is managed by kGraft. For in depth information covering kGraft use, see https://documentation.suse.com/sles/12-SP5/html/SLES-all/cha-kgraft.html.

Before you begin, ensure:

- Uyuni is fully updated.
- You have one or more Salt clients running SLES 12 (SP1 or later).
- Your SLES 12 Salt clients are registered with Uyuni.
- You have access to the SLES 12 channels appropriate for your architecture, including the live patching child channel (or channels).
- The clients are fully synchronized.
- Assign the clients to the cloned channels prepared for live patching. For more information on preparation, see **Administration Guide › Live Patching Channel Setup**.

Procedure: Setting up for Live Patching

1. Select the client you want to manage with Live Patching from **Systems › Overview**, and on the system details page navigate to the **Software › Packages › Install** tab. Search for the kgraft package, and install it.



2. Apply the highstate to enable Live Patching, and reboot the client.
3. Repeat for each client that you want to manage with Live Patching.
4. To check that live patching has been enabled correctly, select the client from **Systems › System List**, and ensure that Live Patching appears in the Kernel field.

Procedure: Applying Live Patches to a Kernel

1. In the Uyuni Web UI, select the client from **Systems › Overview**. A banner at the top of the screen shows the number of critical and non-critical packages available for the client.
2. Click **[Critical]** to see a list of the available critical patches.
3. Select any patch with a synopsis reading Important: Security update for the Linux kernel. Security bugs also include their CVE number, where applicable.
4. OPTIONAL: If you know the CVE number of a patch you want to apply, you can search for it in **Audit › CVE Audit**, and apply the patch to any clients that require it.



- Not all kernel patches are Live Patches. Non-Live kernel patches are represented by a Reboot Required icon located next to the Security shield icon. These patches always require a reboot.
- Not all security issues can be fixed by applying a live patch. Some security issues can only be fixed by applying a full kernel update and require a reboot. The assigned CVE numbers for these issues are not included in live patches. A CVE audit displays this requirement.

Chapter 15. Maintenance Windows

The maintenance windows feature in Uyuni allows you to schedule actions to occur during a scheduled maintenance window period. When you have created your maintenance window schedule, and applied it to a client, you are prevented from executing some actions outside of the specified period.



Maintenance windows operate in a different way to system locking. System locks are switched on or off as required, while maintenance windows define periods of time when actions are allowed. Additionally, the allowed and restricted actions differ. For more information about system locks, see **Client Configuration Guide › System Locking**.

Maintenance windows require both a calendar, and a schedule. The calendar defines the date and time of your maintenance window events, including recurring events, and must be in ical format. The schedule uses the events defined in the calendar to create the maintenance windows. You must create an ical file for upload, or link to an ical file to create the calendar, before you can create the schedule.

When you have created the schedule, you can assign it to clients that are registered to the Uyuni Server. Clients that have a maintenance schedule assigned cannot run restricted actions outside of maintenance windows.

Restricted actions significantly modify the client, and could potentially cause the client to stop running. Some examples of restricted actions are:

- Package installation
- Client upgrade
- Product migration
- Highstate application

Unrestricted actions are minor actions that are considered safe and are unlikely to cause problems on the client. Some examples of unrestricted actions are:

- Package profile update
- Hardware refresh
- Subscribing to software channels

Before you begin, you must create an ical file for upload, or link to an ical file to create the calendar. You can create ical files in your preferred calendaring tool, such as Microsoft Outlook, Google Calendar, or KOrganizer.

Procedure: Uploading a New Maintenance Calendar

1. In the Uyuni Web UI, navigate to **Schedule › Maintenance Windows › Calendars**, and click **[Create]**.
2. In the Calendar Name section, type a name for your calendar.

3. Either provide a URL to your ical file, or upload the file directly.
4. Click **[Create Calendar]** to save your calendar.

Procedure: Creating a New Schedule

1. In the Uyuni Web UI, navigate to **Schedule › Maintenance Windows › Schedules**, and click **[Create]**.
2. In the Schedule Name section, type a name for your schedule.
3. OPTIONAL: If your ical file contains events that apply to more than one schedule, check Multi.
4. Select the calendar to assign to this schedule.
5. Click **[Create Schedule]** to save your schedule.

Procedure: Assigning a Schedule to a Client

1. In the Uyuni Web UI, navigate to **Systems › Systems List**, select the client to be assigned to a schedule, locate the System Properties panel, and click **[Edit These Properties]**. Alternatively, you can assign clients through the system set manager by navigating to **Systems › System Set Manager** and using the **Misc › Maintenance Windows** tab.
2. In the Edit System Details page, locate the Maintenance Schedule field, and select the name of the schedule to be assigned.
3. Click **[Update Properties]** to assign the maintenance schedule.



When you assign a new maintenance schedule to a client, it is possible that the client might already have some restricted actions scheduled, and that these might now conflict with the new maintenance schedule. If this occurs, the Web UI displays an error and you cannot assign the schedule to the client. To resolve this, check the **[Cancel affected actions]** option when you assign the schedule. This cancels any previously scheduled actions that conflict with the new maintenance schedule.

When you have created your maintenance windows, you can schedule restricted actions, such as package upgrades, to be performed during the maintenance window.

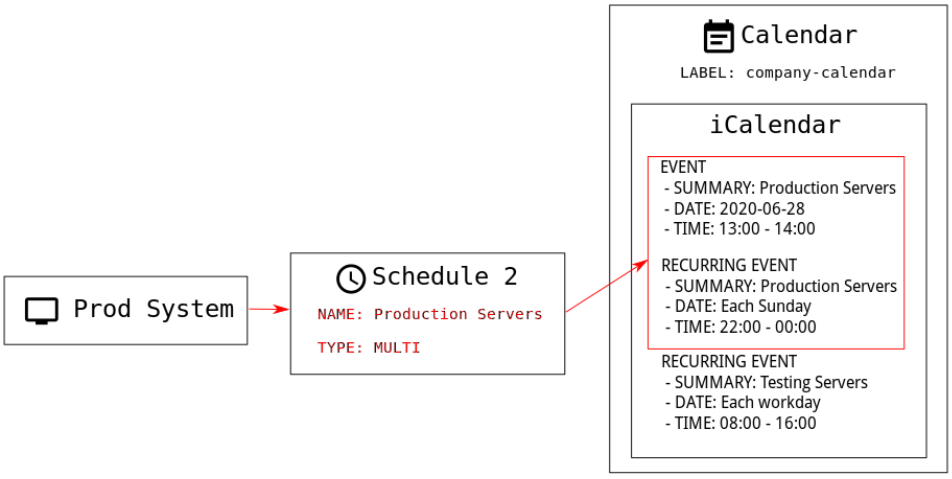
Procedure: Scheduling a Package Upgrade

1. In the Uyuni Web UI, navigate to **Systems › System List**, select the client you want to upgrade, and go to the **Software › Packages › Upgrade** tab.
2. Select the package to upgrade from the list, and click **[Upgrade Packages]**.
3. In the Maintenance Window field, select which maintenance window the client should use to perform the upgrade.
4. Click **[Confirm]** to schedule the package upgrade.

15.1. Maintenance Schedule Types

When you create a calendar, it contains a number of events, which can be either one-time events, or recurring events. Each event contains a summary field. If you want to create multiple maintenance schedules for one calendar, you can specify events for each using the summary field.

For example, you might like to create a schedule for production servers, and a different schedule for testing servers. In this case, you would specify SUMMARY: Production Servers on events for the production servers, and SUMMARY: Testing Servers on events for the testing servers.



There are two types of schedule: single, or multi. If your calendar contains events that apply to more than one schedule, then you must select multi, and ensure you name the schedule according to the summary field you used in the calendar file.

Procedure: Creating a Multi Schedule

1. In the Uyuni Web UI, navigate to **Schedule › Maintenance Windows › Schedules**, and click **[Create]**.
2. In the Schedule Name section, type the name for your schedule. Ensure it matches the summary field of the calendar.
3. Check the Multi option.
4. Select the calendar to assign to this schedule.
5. Click **[Create Schedule]** to save your schedule.
6. To create the next schedule, click **[Create]**.
7. In the Schedule Name section, type the name for your second schedule. Ensure it matches the summary field of the second calendar.
8. Check the Multi option.
9. Click **[Create Schedule]** to save your schedule.

10. Repeat for each schedule you need to create.

15.2. Restricted and Unrestricted Actions

This sections contains a complete list of restricted and unrestricted actions.

Restricted actions significantly modify the client, and could potentially cause the client to stop running. Restricted actions can only be run during a maintenance window. The restricted actions are:

- Package operations (for example, installing, updating, or removing packages)
- Patch updates
- Rebooting a client
- Rolling back transactions
- Configuration management changing tasks
- Applying a highstate
- Autoinstallation and reinstallation
- Remote commands
- Product migrations
- Cluster operations



It is possible to run remote commands directly at any time by navigating to **Salt › Remote Commands**. This applies whether or not the client is in a maintenance window. For more information about remote commands, see **Administration Guide › Actions**.

Unrestricted actions are minor actions that are considered safe and are unlikely to cause problems on the client. If an action is not restricted it is, by definition, unrestricted, and can be run at any time.

Chapter 16. Using mgr-sync

The mgr-sync tool is used at the command prompt. It provides functions for using Uyuni that are not always available in the Web UI. The primary use of mgr-sync is to connect to the SUSE Customer Center, retrieve product and package information, and prepare channels for synchronization with the Uyuni Server.

This tool is designed for use with a SUSE support subscription. It is not required for open source distributions, including openSUSE, CentOS, and Ubuntu.

The available commands and arguments for mgr-sync are listed in this table. Use this syntax for mgr-sync commands:

```
mgr-sync [-h] [--version] [-v] [-s] [-d {1,2,3}] {list,add,refresh,delete}
```

Table 3. mgr-sync Commands

Command	Description	Example Use
list	List channels, organization credentials, or products	mgr-sync list channels
add	Add channels, organization credentials, or products	mgr-sync add channel <channel_name>
refresh	Refresh the local copy of products, channels, and subscriptions	mgr-sync refresh
delete	Delete existing SCC organization credentials from the local system	mgr-sync delete credentials
sync	Synchronize specified channel or ask for it when left blank	mgr-sync sync channel <channel_name>

To see the full list of options specific to a command, use this command:

```
mgr-sync <command> --help
```

Table 4. mgr-sync Optional Arguments

Option	Abbreviated option	Description	Example Use
help	h	Display the command usage and options	mgr-sync --help

Option	Abbreviated option	Description	Example Use
version	N/A	Display the currently installed version of mgr-sync	mgr-sync --version
verbose	v	Provide verbose output	mgr-sync --verbose refresh
store-credentials	s	Store credentials a local hidden file	mgr-sync --store -credentials
debug	d	Log additional debugging information. Requires a level of 1, 2, 3. 3 provides the highest amnount of debugging information	mgr-sync -d 3 refresh
no-sync	N/A	Use with the add command to add products or channels without beginning a synchronization	mgr-sync --no-sync add <channel_name>

Logs for mgr-sync are located in:

- /var/lib/containers/storage/volumes/var-log/_data/rhn/mgr-sync.log
- /var/lib/containers/storage/volumes/var-log/_data/rhn/rhn_web_api.log

Chapter 17. Monitoring with Prometheus and Grafana

You can monitor your Uyuni environment using Prometheus and Grafana. Uyuni Server and Proxy are able to provide self-health metrics. You can also install and manage a number of Prometheus exporters on Salt clients.

17.1. Requirements

Prometheus and Grafana packages are included in the Uyuni Client Tools for:

- SUSE Linux Enterprise 15
- SUSE Linux Enterprise 16
- openSUSE Leap 15.x
- openSUSE Leap 16.x



Only the above listed clients are supported as a monitoring server.

You need to install Prometheus and Grafana on a machine separate from the Uyuni Server. We recommend to use a managed Salt SUSE client as your monitoring server.

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to monitored clients. Clients must have corresponding open ports and be reachable over the network. Alternatively, you can use reverse proxies to establish a connection.

17.2. Prometheus and Grafana

17.2.1. Prometheus

Prometheus is an open-source monitoring tool that is used to record real-time metrics in a time-series database. Metrics are pulled via HTTP, enabling high performance and scalability.

Prometheus metrics are time series data, or timestamped values belonging to the same group or dimension. A metric is uniquely identified by its name and set of labels.

```
   metric name              labels                 timestamp     value
 ┌──────────────┴──────────────┐   ┌──────────────────┴──────────────────┐
 ┌──────────┴──────────┐   ┌──┴──┐
http_requests_total{status="200", method="GET"}  @1557331801.111  42236
```

Each application or system being monitored must expose metrics in the format above, either through code instrumentation or Prometheus exporters.

17.2.2. Prometheus exporters

Exporters are libraries that help with exporting metrics from third-party systems as Prometheus metrics. Exporters are useful whenever it is not feasible to instrument a given application or system with Prometheus metrics directly. Multiple exporters can run on a monitored host to export local metrics.

The Prometheus community provides a list of official exporters, and more can be found as community contributions. For more information and an extensive list of exporters, see https://prometheus.io/docs/instrumenting/exporters/.

17.2.3. Grafana

Grafana is a tool for data visualization, monitoring, and analysis. It is used to create dashboards with panels representing specific metrics over a set period of time. Grafana is commonly used together with Prometheus, but also supports other data sources such as ElasticSearch, MySQL, PostgreSQL, and Influx DB. For more information about Grafana, see https://grafana.com/docs/.

17.3. Set up the monitoring server

To set up your monitoring server, you need to install Prometheus and Grafana, and configure them.



Only SUSE clients are supported as a monitoring server. For a complete list, see **Administration Guide › Monitoring › Monitoring Requirements**.

17.3.1. Install Prometheus

If your monitoring server is a Salt client, you can install the Prometheus package using the Uyuni Web UI. Otherwise you can download and install the package on your monitoring server manually. The Prometheus software is also available for Uyuni Proxy and Uyuni for Retail Branch Server.



- To access a shell inside the Uyuni Server container run `mgrctl term` on the container host, or to execute one command run `mgrctl exec <options> — <command>`.
- To copy files from inside the container to the container host use `mgrctl cp`.



Prometheus expects POSIX filesystem for storing data. Non-POSIX compliant filesystems are not supported. NFS filesystems are not supported.

Procedure: Installing Prometheus using the Web UI

1. In the Uyuni Web UI, open the details page of the system where Prometheus is to be installed, and navigate to the Formulas tab.

2. Check the Prometheus checkbox to enable monitoring formulas, and click [Save].
3. Navigate to the Prometheus tab in the top menu.
4. In the Uyuni Server section, enter valid Uyuni API credentials. Make sure that the credentials you have entered allow access to the set of systems you want to monitor.
5. Customize any other configuration options according to your needs.
6. Click [Save Formula].
7. Apply the highstate and confirm that it completes successfully.
8. Check that the Prometheus interface loads correctly. In your browser, navigate to the URL of the server where Prometheus is installed, on port 9090 (for example, http://example.com:9090).

For more information about the monitoring formulas, see **Specialized Guides › Salt › Salt Formula Monitoring**.

Procedure: Manually installing and configuring Prometheus

1. On the monitoring server, install the golang-github-prometheus-prometheus package:

```
zypper in golang-github-prometheus-prometheus
```

2. Enable the Prometheus service:

```
systemctl enable --now prometheus
```

3. Check that the Prometheus interface loads correctly. In your browser, navigate to the URL of the server where Prometheus is installed, on port 9090 (for example, http://example.com:9090).
4. Open the configuration file at /etc/prometheus/prometheus.yml and add this configuration information. Replace server.url with your Uyuni server URL and adjust username and password fields to match your Uyuni credentials.

```
# {productname} self-health metrics
scrape_configs:
- job_name: 'mgr-server'
  static_configs:
    - targets:
      - 'server.url:9100'  # Node exporter
      - 'server.url:9187'  # PostgreSQL exporter
      - 'server.url:5556'  # JMX exporter (Tomcat)
      - 'server.url:5557'  # JMX exporter (Taskomatic)
      - 'server.url:9800'  # Taskomatic
    - targets:
      - 'server.url:80'    # Message queue
      labels:
        __metrics_path__: /rhn/metrics

# Managed systems metrics:
- job_name: 'mgr-clients'
  uyuni_sd_configs:
   - server: "http://server.url"
     username: "admin"
     password: "admin"
  relabel_configs:
   - source_labels: [__meta_uyuni_exporter]
     target_label: exporter
   - source_labels: [__address__]
     replacement: "No group"
     target_label: groups
   - source_labels: [__meta_uyuni_groups]
     regex: (.+)
     target_label: groups
   - source_labels: [__meta_uyuni_minion_hostname]
     target_label: hostname
   - source_labels: [__meta_uyuni_primary_fqdn]
     regex: (.+)
     target_label: hostname
   - source_labels: [hostname, __address__]
     regex: (.*);.*:(.*)
     replacement: ${1}:${2}
     target_label: __address__
   - source_labels: [__meta_uyuni_metrics_path]
     regex: (.+)
     target_label: __metrics_path__
   - source_labels: [__meta_uyuni_proxy_module]
     target_label: __param_module
   - source_labels: [__meta_uyuni_scheme]
     target_label: __scheme__
```

5. Save the configuration file.
6. Restart the Prometheus service:

```
systemctl restart prometheus
```

For more information about the Prometheus configuration options, see the official Prometheus documentation at https://prometheus.io/docs/prometheus/latest/configuration/configuration/.

17.3.2. Install Grafana

If your monitoring server is a client managed by Uyuni, you can install the Grafana package using the Uyuni Web UI. Otherwise you can download and install the package on your monitoring server manually.

Grafana is not available on Uyuni Proxy.

Procedure: Installing Grafana using the Web UI

1. In the Uyuni Web UI, open the details page of the system where Grafana is to be installed, and navigate to the Formulas tab.
2. Check the Grafana checkbox to enable monitoring formulas, and click **[Save]**.
3. Navigate to the Grafana tab in the top menu.
4. In the Enable and configure Grafana section, enter the admin credentials you want to use to log in Grafana.
5. On the Datasources section, make sure that the Prometheus URL field points to the system where Prometheus is running.
6. Customize any other configuration options according to your needs.
7. Click **[Save Formula]**.
8. Apply the highstate and confirm that it completes successfully.
9. Check that the Grafana interface is loading correctly. In your browser, navigate to the URL of the server where Grafana is installed, on port 3000 (for example, http://example.com:3000).
    - Ensure that port 3000 is open on the firewall to successfully access Grafana

Uyuni provides pre-built dashboards for server self-health, basic client monitoring, and more. You can choose which dashboards to provision in the formula configuration page.

Procedure: Manually installing Grafana

1. Install the grafana package:

```
zypper in grafana
```

2. Enable the Grafana service:

```
systemctl enable --now grafana-server
```

3. In your browser, navigate to the URL of the server where Grafana is installed, on port 3000 (for example, http://example.com:3000).
    - Ensure that port 3000 is open on the firewall to successfully access Grafana.
4. On the login page, enter admin for username and password.
5. Click **[Log in]**. If login is successful, then you will see a prompt to change the password.
6. Click **[OK]** on the prompt, then change your password.
7. Move your cursor to the cog icon on the side menu which will show the configuration options.
8. Click **[Data sources]**.
9. Click **[Add data source]** to see a list of all supported data sources.
10. Choose the Prometheus data source.
11. Make sure to specify the correct URL of the Prometheus server.
12. Click **[Save & test]**.
13. To import a dashboard click the **[+]** icon in the side menu, and then click **[Import]**.
14. For Uyuni server overview load the dashboard ID: 17569.
15. For Uyuni clients overview load the dashboard ID: 17570.





- For more information about the monitoring formulas, see **Specialized Guides › Salt › Salt Formula Monitoring**.
- For more information on how to manually install and configure Grafana, see https://grafana.com/docs.

17.4. Configure Uyuni monitoring

With Uyuni 4 and higher, you can enable the server to expose Prometheus self-health metrics, and also install and configure exporters on client systems.

17.4.1. Server self-monitoring

The Server self-health metrics cover hardware, operating system and Uyuni internals. These metrics are made available by instrumentation of the Java application, combined with Prometheus exporters.

These exporters are shipped with Uyuni Server:

- Node exporter: golang-github-prometheus-node_exporter.
  - See https://github.com/prometheus/node_exporter.
- PostgreSQL exporter: prometheus-postgres_exporter.
  - See https://github.com/wrouesnel/postgres_exporter.
- JMX exporter: prometheus-jmx_exporter.
  - See https://github.com/prometheus/jmx_exporter.

These exporter packages are shipped with Uyuni Proxy:

- Node exporter: golang-github-prometheus-node_exporter.
  - See https://github.com/prometheus/node_exporter.

- Squid exporter: golang-github-boynux-squid_exporter.
  - See https://github.com/boynux/squid-exporter.

The exporters are pre-installed in Uyuni Server and Proxy, but their respective systemd daemons are disabled by default.

Procedure: Enabling self-monitoring

1. In the Uyuni Web UI, navigate to **Admin › Manager Configuration › Monitoring**.
2. Click **[Enable services]**.
3. Restart Tomcat and Taskomatic.
4. Navigate to the URL of your Prometheus server, on port 9090 (for example, http://example.com:9090)
5. In the Prometheus UI, navigate to **Status › Targets** and confirm that all the endpoints on the mgr-server group are up.
6. If you have also installed Grafana with the Web UI, the server insights are visible on the Uyuni Server dashboard, in the **Admin › Manager Configuration › Monitoring**.



Only server self-health monitoring can be enabled using the Web UI. Metrics for a proxy are not automatically collected by Prometheus. To enable self-health monitoring on a proxy, you need to manually install exporters and enable them.

The following relevant metrics are collected on the Uyuni Server.

Table 5. Server statistics (port 80)

Metric	Type	Description
uyuni_all_systems	gauge	Number of all systems
uyuni_virtual_systems	gauge	Number of virtual systems
uyuni_inactive_systems	gauge	Number of inactive systems
uyuni_outdated_systems	gauge	Number of systems with outdated packages

Table 6. PostgreSQL exporter (port 9187)

Metric	Type	Description
pg_stat_database_tup_fetched	counter	Number of rows fetched by queries
pg_stat_database_tup_inserted	counter	Number of rows inserted by queries
pg_stat_database_tup_updated	counter	Number of rows updated by queries
pg_stat_database_tup_deleted	counter	Number of rows deleted by queries
mgr_serveractions_completed	gauge	Number of completed actions
mgr_serveractions_failed	gauge	Number of failed actions
mgr_serveractions_picked_up	gauge	Number of picked-up actions
mgr_serveractions_queued	gauge	Number of queued actions

Table 7. JMX exporter (Tomcat port 5556, Taskomatic port 5557)

Metric	Type	Description
java_lang_Threading_ThreadCount	gauge	Number of active threads
java_lang_Memory_HeapMemoryUsage_used	gauge	Current heap memory usage

Table 8. Server Message Queue (port 80)

Metric	Type	Description
message_queue_thread_pool_threads	counter	Number of message queue threads ever created
message_queue_thread_pool_threads_active	gauge	Number of currently active message queue threads
message_queue_thread_pool_task_count	counter	Number of tasks ever submitted
message_queue_thread_pool_completed_task_count	counter	Number of tasks ever completed

Table 9. Salt Queue (port 80)

Metric	Type	Description
salt_queue_thread_pool_size	gauge	Number of threads created per Salt queue

Metric	Type	Description
salt_queue_thread_pool_active_threads	gauge	Number of currently active Salt threads per queue
salt_queue_thread_pool_task_total	counter	Number of tasks ever submitted per queue
salt_queue_thread_pool_completed_task_total	counter	Number of tasks ever completed per queue

Every salt_queue value has a label named queue with the queue number as value.

Table 10. Taskomatic Scheduler (port 9800)

Metric	Type	Description
taskomatic_scheduler_threads	counter	Number of scheduler threads ever created
taskomatic_scheduler_threads_active	gauge	Number of currently active scheduler threads
taskomatic_scheduler_completed_task_count	counter	Number of tasks ever completed

17.4.2. Monitoring managed systems

Prometheus metrics exporters can be installed and configured on Salt clients using formulas. The packages are available from the Uyuni client tools channels, and can be enabled and configured directly in the Uyuni Web UI.

These exporters can be installed on managed systems:

- Node exporter: golang-github-prometheus-node_exporter.
  - See https://github.com/prometheus/node_exporter.
- PostgreSQL exporter: prometheus-postgres_exporter.
  - See https://github.com/wrouesnel/postgres_exporter.
- Apache exporter: golang-github-lusitaniae-apache_exporter.
  - See https://github.com/Lusitaniae/apache_exporter.



On SL Micro, only the Node exporter and the Blackbox exporter are available.

When you have the exporters installed and configured, you can start using Prometheus to collect metrics from the monitored systems. If you have configured your monitoring server with the Web UI, metrics collection happens automatically.

Procedure: Configuring Prometheus exporters on a client

1. In the Uyuni Web UI, open the details page of the client to be monitored, and navigate to the **Formulas** tab.
2. Check the Enabled checkbox on the Prometheus Exporters formula.
3. Click **[Save]**.
4. Navigate to the **Formulas › Prometheus Exporters** tab.
5. Select the exporters you want to enable and customize arguments according to your needs. The Address field accepts either a port number preceded by a colon (:9100), or a fully resolvable address (example:9100).
6. Click **[Save Formula]**.
7. Apply the highstate.



The following ports need to be enabled and not blocked by the firewall:

- on the monitor: 9090/tcp
- on the monitored system: 9100/tcp, 9117/tcp, and 9187/tcp



Monitoring formulas can also be configured for System Groups, by applying the same configuration used for individual systems inside the corresponding group.

For more information about the monitoring formulas, see **Specialized Guides › Salt › Salt Formula Monitoring**.

17.4.3. Change Grafana password

To change the Grafana password follow the steps described in the Grafana documentation:

- https://grafana.com/docs/grafana/latest/administration/user-management/user-preferences/#change-your-grafana-password

In case you have lost the Grafana administrator password you can reset it as root with the following command:

```
grafana-cli --configOverrides cfg:default.paths.data=/var/lib/grafana --homepath
/usr/share/grafana admin reset-admin-password <new_password>
```

17.5. Network boundaries

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to monitored clients. By default, Prometheus uses these ports:

- Node exporter: 9100
- PostgreSQL exporter: 9187
- Apache exporter: 9117

Additionally, if you are running the alert manager on a different host than where you run Prometheus, you also need to open port 9093. The alert manager is part of Prometheus solution. It handles alerts sent by client applications such as the Prometheus server instance. For more information about the alert manager, see https://prometheus.io/docs/alerting/latest/alertmanager/.

For clients installed on cloud instances, you can add the required ports to a security group that has access to the monitoring server.

Alternatively, you can deploy a Prometheus instance in the exporters' local network, and configure federation. This allows the main monitoring server to scrape the time series from the local Prometheus instance. If you use this method, you only need to open the Prometheus API port, which is 9090.

For more information on Prometheus federation, see https://prometheus.io/docs/prometheus/latest/federation/.

You can also proxy requests through the network boundary. Tools like PushProx deploy a proxy and a client on both sides of the network barrier and allow Prometheus to work across network topologies such as NAT.

For more information on PushProx, see https://github.com/RobustPerception/PushProx.

17.5.1. Reverse proxy setup

Prometheus fetches metrics using a pull mechanism, so the server must be able to establish TCP connections to each exporter on the monitored clients. To simplify your firewall configuration, you can use reverse proxy for your exporters to expose all metrics on a single port.

Procedure: Installing Prometheus exporters with reverse proxy

1. In the Uyuni Web UI, open the details page of the system to be monitored, and navigate to the Formulas tab.
2. Check the Prometheus Exporters checkbox to enable the exporters formula, and click **[Save]**.

3. Navigate to the Prometheus Exporters tab in the top menu.
4. Check the Enable reverse proxy option, and enter a valid reverse proxy port number. For example, 9999.
    - Ensure that port 9999/tcp is open on the firewall.
5. Customize the other exporters according to your needs.
6. Click [Save Formula].
7. Apply the highstate and confirm that it completes successfully.

For more information about the monitoring formulas, see **Specialized Guides › Salt › Salt Formula Monitoring**.

17.6. Security

Prometheus server and Prometheus node exporter offer a built-in mechanism to secure their endpoints with TLS encryption and authentication. Uyuni Web UI simplifies the configuration of all involved components. The TLS certificates have to be provided and deployed by the user. Uyuni offers enabling the following security model:

- Node exporter: TLS encryption and client certificate based authentication
- Prometheus: TLS encryption and basic authentication

For more information about configuring all available options, see **Specialized Guides › Salt › Salt Formula Monitoring**.

17.6.1. Generating TLS certificates

By default, Uyuni does not provide any certificates for securing monitoring configuration. For providing security, you can generate or import custom certificates, self-signed or signed by a third party certificate authority (CA).

This section demonstrates how to generate client/server certificates for Prometheus and Node exporter minions self-signed with Uyuni CA.

Procedure: Creating server/client TLS certificate

1. At the command prompt of the Uyuni container host, as root, run the following commands:

a. To generate certificate files, run the following command.

Ensure that the set-cname parameter is the fully qualified domain name (FQDN) of your Salt client. You can use the set-cname parameter multiple times if you require multiple aliases:

```
mgrctl exec -ti -- mgr-ssl-tool --gen-server --dir="/root/ssl-build" --set
-country="COUNTRY" \
  --set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
  --set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
  --set-hostname="minion.example.com" --set-cname="minion.example.com" --no
-rpm
```

Resulting in:

```
Generating the web server's SSL private key: /root/ssl-
build/minion/server.key
Generating web server's SSL certificate request: /root/ssl-
build/minion/server.csr
Generating/signing web server's SSL certificate: server.crt
```

b. Copy server.crt and server.key files from the server container to the host:

```
mgrctl cp server:/root/ssl-build/minion/server.key server.key
mgrctl cp server:/root/ssl-build/minion/server.crt server.crt
```

c. Copy server.crt and server.key files from the host to the monitoring client:

```
ssh minion.example.com 'mkdir /etc/ssl/mlm-server-certs'
scp /root/server.* minion.example.com:/etc/ssl/mlm-server-certs
ssh minion.example.com 'chmod go+r /etc/ssl/mlm-server-certs/server.*; ls
-la /etc/ssl/mlm-server-certs'
```

2. To configure Salt formulars, enter the directory names specified in the previous steps.

a. formular server

```
Server Certificate /etc/ssl/mlm-server-certs/server.crt
Server Key /etc/ssl/mlm-server-certs/server.key
```

b. formular minion

Chapter 18. Organizations

Organizations are used to manage user access and permissions within Uyuni.

For most environments, a single organization is enough. However, more complicated environments might need several organizations. You might like to have an organization for each physical location within your business, or for different business functions.

When you have created your organizations, you can create and assign users to your organizations. You can then assign permissions on an organization level, which applies by default to every user assigned to the organization.

You can also configure authentication methods for your new organization, including PAM and single sign-on. For more information about authentication, see **Administration Guide › Auth Methods**.



You must be logged in as the Uyuni administrator to create and manage organizations.

Procedure: Creating a New Organization

1. In the Uyuni Web UI, navigate to **Admin › Organizations**, and click **[Create Organization]**.
2. In the Create Organization dialog, complete these fields:
    - In the Organization Name field, type a name for your new organization. The name should be between 3 and 128 characters long.
    - In the Desired Login field, type the login name you want to use for the organization's administrator. This must be a new administrator account, you are not be able to use an existing administrator account to sign in to the new organization, including the one you are currently signed in with.
    - In the Desired Password field, type a password for the new organization's administrator. Confirm the password by typing it again in the Confirm Password field. Password strength is indicated by the colored bar beneath the password fields.
    - In the Email field, type an email address for the new organization's administrator.
    - In the First Name field, select a salutation, and type a given name for the new organization's administrator.
    - In the Last Name field, type a surname for the new organization's administrator.
3. Click **[Create Organization]**.

18.1. Manage Organizations

In the Uyuni Web UI, navigate to **Admin › Organizations** to see a list of available organizations. Click the name of an organization to manage it.

From the **Admin › Organizations** section, you can access tabs to manage users, trusts, configuration, and states

for your organization.



Organizations can only be managed by their administrators. To manage an organization, ensure you are signed in as the correct administrator for the organization you want to change.

18.1.1. Organization Users

Navigate to the Users tab to view the list of all users associated with the organization, and their role. Clicking a username takes you to the Users menu to add, change, or delete users.

18.1.2. Trusted Organizations

Navigate to the Trusts tab to add or remove trusted organizations. Establishing trust between organizations allow them to share content between them, and gives you the ability to transfer clients from one organization to another.

18.1.3. Configure Organizations

Navigate to the Configuration tab to manage the configuration of your organization. This includes the use of staged contents, and the use of SCAP files.

For more information about content staging, see **Administration Guide › Content Staging**.

For more information about OpenSCAP, see **Reference Guide › Audit › Audit Openscap Overview**.

18.2. Manage States

Navigate to the States tab to manage Salt states for all clients in your organization. States allow you to define global security policies, or add a common admin user to all clients.

For more information about Salt States, see **Specialized Guides › Salt › Salt States and Pillars**.

18.2.1. Manage Configuration Channels

You can select which configuration channels should be applied across your organization. Configuration channels can be created in the Uyuni Web UI by navigating to **Configuration › Channels**. Apply configuration channels to your organization using the Uyuni Web UI.

Procedure: Applying Configuration Channels to an Organization

1. In the Uyuni Web UI, navigate to **Home › My Organization › Configuration Channels**.

2. Use the search feature to locate a channel by name.
3. Check the channel to be applied and click **[Save Changes]**. This saves to the database, but does not apply the changes to the channel.
4. Apply the changes by clicking **[Apply]**. This schedules the task to apply the changes to all clients within the organization.

Chapter 19. Patch Management

This chapter contains various topics related to patch management.

19.1. Retracted Patches

When a new patch gets released by the vendor, it might happen that the patch has undesirable side effects (security, stability) in some scenario that was not identified by testing. When this happens (very rarely), vendors typically release a new patch, which may take hours or days, depending on the internal processes in place by that vendor.

SUSE has introduced a new mechanism (2021) called "retracted patches" to revoke such patches almost immediately by setting their advisory status to "retracted" (instead of "final" or "stable").



A patch is "retracted," when its advisory status attribute is set to "retracted." A package is "retracted," when it belongs to a "retracted" patch.

A retracted patch or package cannot be installed on systems with Uyuni. The only way to install a retracted package, is to do it manually with zypper install and specifying the exact package version. For example:

```
zypper install vim-8.0.1568-5.14.1
```

Retracted status of patches and packages is depicted with the ⊗ icon in the Web UI of Uyuni. For example, see:

- list of packages in a channel
- list of patches in a channel

When a patch or package, that has been installed on a system, gets retracted, the ⊗ icon is also displayed in the installed packages list of that system. Uyuni does not provide a way to downgrade such a patch or package.

19.1.1. Channel Clones

When using cloned channels, you must pay attention to the propagation of the retracted advisory status from the original channels to the clones.

Upon cloning vendor channels into your organization, channel patches will be cloned as well.

When the vendor retracts a patch in a channel and Uyuni synchronizes this channel (for example, with the nightly job), the "retracted" attribute will not get propagated to the cloned patches and will not be observed by the clients subscribed to cloned channels. To propagate the attribute to your cloned channels use one of the following ways:

- Patch Sync (**Software › Manage › cloned channel › Patches › Sync**). This function allows you to align the attributes of patches in your cloned channel to their originals.
- Content Lifecycle Management. For more information about cloned channels in the context of Content Lifecycle Management, see **Client Configuration Guide › Channels**.

19.1.2. Patch sharing

When you create multiple vendor channel clones in your organization, the patches are not cloned multiple times, but are shared between cloned channels. As a consequence, when you synchronize your cloned patch (either using the patch sync function or with Content Lifecycle Management mentioned above), all channels using the cloned patch will observe that change.

Example:

1. Consider two Content Lifecycle Management projects `prj1` and `prj2`
2. Both of these projects have 2 environments `dev` and `test`
3. Both of these projects have a vendor channel set as a source channel
4. All channels in this scenario (four cloned channels in total) are aligned to the latest state of the vendor channels
5. Vendor retracts a patch in the source channel and the nighly job synchronizes it to your Uyuni
6. None of the four channels see this change because they are using a patch clone, not the patch directly.
7. As soon as you synchronize your patch (either you build any of these two projects, or you use the Patch Sync function on any of the four cloned channels), due to the patch sharing, **ALL** of the cloned channels will see the patch as retracted.

Chapter 20. Using PTFs in Uyuni

SUSE provides temporary fixes for all currently supported solutions shipped directly to its customers. These PTFs (Program Temporary Fixes) are now available as repositories, which can be synced in Uyuni.

20.1. Understanding PTF packages

PTF packages are installed via a proxy package and are named ptf-xxxxxx, where xxxxxx is a number and part of the name of the package, not its version.

They will depend on the correct version of the package that is known to include the correction in the software. This type of package:

- cannot be installed accidentally (i.e. zypper update will never suggest installing them),
- cannot be removed accidentally (i.e. a newer package version will not replace the PTF one, unless the user makes it explicitly on the zypper command line),
- is only updated when the newer version is known to address the specific issue previously solved by the PTF,
- updates only packages already installed on the system (i.e. if a software is split into multiple packages, the PTF will replace only those currently installed on the system).

The correct ID of the package will be provided by SUSE Support during the course of the support case investigation, along with instruction on how to deploy/restart the affected services.

20.2. Installing PTF packages



Access to PTF channels via Uyuni needs to be requested from L3 support.

Procedure: Enabling and synchronizing PTF repositories using the command line

1. On the console enter mgr-sync refresh.
2. Enter mgr-sync list channel and look for channels starting with your SCC account name and ptfs in its name. For example, a123456-sles-15.3-ptfs-x86_64.
3. Enable the PTF channel with mgr-sync add channel <label>.

This channel is now available and can be added to every system which is using the same base channel.

PTF packages need to be installed explicitly, since they are not automatically picked up when updating a system. The SUSE Customer Support will provide the PTF number to fix a specific problem. With the number

the proxy package can be identified in the PTF list. In Uyuni Web UI every system with PTFs available for installation has a page which lists them.

Procedure: Enabling and synchronizing PTF repositories via the Uyuni Web UI

1. In the Uyuni Web UI, navigate to **Admin › Setup Wizard › Products** and look for the product you want to enable the PTF repository for.
2. Click **[Show product's channels]** next to the products sync status.
3. You should see a popup listing mandatory and optional channels for the product.
4. In the optional channels list look for channels starting with your SCC account name and ptfs in its name. For example a123456-sles-15.3-ptfs-x86_64.
5. Select the channel using the checkbox next to its name and click **[Confirm]** to schedule the sync.

Note that the product has to be installed to be able to add optional channels to it.

Procedure: Installing the PTF packages

1. In the Uyuni Web UI, navigate to **Systems › Systems List** and select the client where you want to install a PTF.
2. Navigate to the **Systems › Software › Packages › Software Channels** and select the PTF channel.
3. Click **[Next]**, and Confirm Software Channel Change with **[Confirm]**.
4. To check if the channel assignment is finished, navigate to **Systems › Events › History** to see the results.
5. Navigate to the **Systems › Software › PTFs › Install** sub tab.
6. Select the PTF package you want to install.
7. Click **[Install PTF]**, and Confirm Program Temporary Fixes (PTFs) Installation with **[Confirm]**.

8. To see PTF installation results, navigate to **Systems › Events › History**.

In case a PTF should be installed using the API, the normal system.schedulePackageInstall API can be used with the proxy package name.

20.3. After PTF installation

Once a PTF is confirmed to address the reported issue, the updated package will be tracked for inclusion into a future maintenance update before being widely distributed as an regular maintenance update in the update repositories.

When this regular update with the fix is released, an updated version of the PTF will also be released into the account-specific PTF repository. The updated PTF will remove the strict dependencies and allow updates to be installed again.

The replacement of the PTF with the maintenance update which includes the fix happens automatically via a standard package update or patch installation.

20.4. Removing the patched version of a package

In cases where a PTF needs to be uninstalled and the non-patched version of the packages need to be installed on the system, a simple package remove cannot be used. The PTF package is not selectable in the standard package list page.

Procedure: Removing the PTF packages

1. In the Uyuni Web UI, navigate to **Systems › Systems List** and select the client you want to remove a PTF from.
2. Navigate to the **Systems › Software › PTFs › List/Remove** sub tab.
3. Select the ptf package you want to remove.
4. Click **[Remove PTFs]**, and on the Confirm Program Temporary Fixes (PTFs) Removal page click **[Confirm]**.
5. To see the results, navigate to **Systems › Events › History**.



Removing PTFs requires a special version of libzypp and zypper to be installed on the client system. Check zypper --help to confirm whether removeptf is supported. The List/Remove tab is only visible if this condition is met.

In case the PTF should be removed using the API, the normal system.schedulePackageRemove API can be used

with the proxy package name.

20.5. Removing the patched version of a package on the client

In case a PTF should be removed directly on the client using the console, it is required to use a special command zypper removeptf. All other ways result either in an error or can lead to unwanted behavior like removing important packages from the system and make the system unusable.

Chapter 21. Generate Reports

Uyuni allows the user to produce a variety of reports. These reports can be helpful for taking inventory of your subscribed systems, users, and organizations. Using reports is often simpler than gathering information manually from the Uyuni Web UI, especially if you have many systems under management.

While the command line tool spacewalk-report can be used to generate pre-configured reports, with the introduction of the **Administration Guide › Reporting Database** it is also possible to generate fully customized reports. This can be achieved by connecting any reporting tool that supports the SQL language to the reporting database and extract the data directly. For more information about the data availability and structure, see the reporting database schema documentation.

21.1. Using spacewalk-report



Use mgrctl term before running steps inside the server container.

To generate reports, you must have the spacewalk-reports package installed. The spacewalk-report command allows you to organize and display reports about content, systems, and user resources across Uyuni.



Due to the introduction of the **Administration Guide › Reporting Database**, spacewalk-report now gathers by default the data from the reporting database. See spacewalk-report and the reporting database for more information.

You can generate reports on:

System Inventory

List all the systems registered to Uyuni.

Patches

List all the patches relevant to the registered systems. You can sort patches by severity, as well as the systems that apply to a particular patch.

Users

List all registered users and any systems associated with a particular user.

To get the report in CSV format, run this command at the command prompt on the server:

```
spacewalk-report <report_name>
```

21.2. spacewalk-report and the reporting database

spacewalk-report uses by default the new reporting database to extract the data. This means that the new

generated reports will have some differences in the structure and the format of the data. The differences common to all reports are:

- the report data is not changing in real-time, but it's updated only by the execution of a scheduled task;
- data duplication has been removed and columns that were previously considered "multival" contain now multiple values separated by ;. This also means that the command line options --multival-on-rows and --multival-separator are no longer applicable to the new reports, as their behavior is now the default;
- in all reports the new columns mgm_id and synced_date have been introduced to identify the management server in the hub scenario and the last time the information was updated from the application database;
- all boolean values are now represented by the True/False and not by a 1/0 values;
- the column org_id has been replaced by organization, which contains the organization name and not the numerical identifier;
- The term "server" has been replaced by "system." So, for example, the column server_id is now called system_id.

For report specific changes, see List of available reports.



If this changed behavior causes trouble, the new option --legacy-report can be used to fall back to the old report, which is executed against the application database.

For more information about the reporting database, see **Administration Guide › Reporting Database**. For hub-specific aggregation of data from multiple servers, see **Specialized Guides › Large Deployments › Hub Reporting**.

21.3. List of available reports

This table lists the available reports:

Table 11. spacewalk-report Reports

Report	Invoked as	Description	Uses reporting database	Specific differences
Actions	actions	All actions.	Yes	The column id is now called action_id

Report	Invoked as	Description	Uses reporting database	Specific differences
Activation Keys	activation-keys	All activation keys, and the entitlements, channels, configuration channels, system groups, and packages associated with them.	No	
Activation Keys: Channels	activation-keys-channels	All activation keys and the entities associated with each key.	No	
Activation Keys: Configuration	activation-keys-config	All activation keys and the configuration channels associated with each key.	No	
Activation Keys: Server Groups	activation-keys-groups	All activation keys and the system groups associated with each key.	No	
Activation Keys: Packages	activation-keys-packages	All activation keys and the packages each key can deploy.	No	
Channel Packages	channel-packages	All packages in a channel.	Yes	
Channel Report	channels	Detailed report of a given channel.	Yes	
Cloned Channel Report	cloned-channels	Detailed report of cloned channels.	Yes	

Report	Invoked as	Description	Uses reporting database	Specific differences
Configuration Files	config-files	All configuration file revisions for all organizations, including file contents and file information.	No	
Latest Configuration Files	config-files-latest	The most recent configuration file revisions for all organizations, including file contents and file information.	No	
Custom Channels	custom-channels	Channel metadata for all channels owned by specific organizations.	Yes	The column id is now called channel_id
Custom Info	custom-info	Client custom information.	Yes	
Patches in Channels	errata-channels	All patches in channels.	Yes	
Patches Details	errata-list	All patches that affect registered clients.	Yes	
All patches	errata-list-all	All patches.	No	
Patches for Clients	errata-systems	Applicable patches and any registered clients that are affected.	Yes	
Host Guests	host-guests	Host and guests mapping.	Yes	
Inactive Clients	inactive-systems	Inactive clients.	Yes	The mandatory parameter is now called threshold.

Report	Invoked as	Description	Uses reporting database	Specific differences
System Inventory	inventory	Clients registered to the server, together with hardware and software information.	Yes	The column osad_status has been removed.
Kickstart Scripts	kickstart-scripts	All kickstart scripts, with details.	No	
Kickstart Trees	kickstartable-trees	Kickstartable trees.	No	"
All Upgradable Versions	packages-updates-all	All newer package versions that can be upgraded.	Yes	
Newest Upgradable Version	packages-updates-newest	Newest package versions that can be upgraded.	Yes	
Proxy Overview	proxies-overview	All proxies and the clients registered to each.	Yes	
Repositories	repositories	All repositories, with their associated SSL details, and any filters.	No	
Result of SCAP	scap-scan	Result of OpenSCAP sccdf evaluations.	Yes	
Result of SCAP	scap-scan-results	Result of OpenSCAP sccdf evaluations, in a different format.	Yes	
System Data	splice-export	Client data needed for splice integration.	No	
System Currency	system-currency	Number of available patches for each registered client.	No	

Report	Invoked as	Description	Uses reporting database	Specific differences
System Extra Packages	system-extra-packages	All packages installed on all clients that are not available from channels the client is subscribed to.	Yes	
System Groups	system-groups	System groups.	Yes	
Activation Keys for System Groups	system-groups-keys	Activation keys for system groups.	No	
Systems in System Groups	system-groups-systems	Clients in system groups.	Yes	
System Groups Users	system-groups-users	System groups and users that have permissions on them.	No	
System Hardware	system-hardware	System hardware information.	No	
History: System	system-history	Event history for each client.	Yes	
History: Channels	system-history-channels	Channel event history.	Yes	
History: Configuration	system-history-configuration	Configuration event history.	Yes	The column created_date has been removed.
History: Entitlements	system-history-entitlements	System entitlement event history.	Yes	
History: Errata	system-history-errata	Errata event history.	Yes	The column created_date has been removed.
History: Kickstart	system-history-kickstart	Kickstart event history.	Yes	The column created_date has been removed.

Report	Invoked as	Description	Uses reporting database	Specific differences
History: Packages	system-history-packages	Package event history.	Yes	The column created_date has been removed.
History: SCAP	system-history-scap	OpenSCAP event history.	Yes	The column created_date has been removed.
MD5 Certificates	system-md5-certificates	All registered clients using certificates with an MD5 checksum.	No	
Installed Packages	system-packages-installed	Packages installed on clients.	Yes	
System Profiles	system-profiles	All clients registered to the server, with software and system group information.	No	
Users	users	All users registered to Uyuni.	Yes	The column organization_id has been removed.
MD5 Users	users-md5	All users for all organizations using MD5 encrypted passwords, with their details and roles.	Yes	The column organization_id has been removed.
Systems administered	users-systems	Clients that individual users can administer.	Yes	The column organization_id has been removed.

For more information about an individual report, run spacewalk-report with the option --info or --list-fields -info and the report name. This shows the description and list of possible fields in the report.

For further information on program invocation and options, see the spacewalk-report(8) man page as well as the --help parameter of the spacewalk-report command.

21.4. Reporting Database

Uyuni includes a reporting database on every server. The reporting database is a separate PostgreSQL database that stores a de-normalized, read-optimized snapshot of the most relevant data from the main Uyuni database. External reporting tools can connect directly to this database to query and visualize data.

In a hub deployment, the hub aggregates the reporting databases of all connected peripheral servers into a single central reporting database. For hub-specific setup and aggregation, see **Specialized Guides › Large Deployments › Hub Reporting**.

21.4.1. Setup

The reporting database and schema are set up by default using the local PostgreSQL server. The reporting database is a separate database accessible via the network.



In a hub deployment, the connection between hub and peripheral servers is established via hub online synchronization. For more information, see **Specialized Guides › Large Deployments › Hub Online Sync**.

21.4.1.1. Create a DB User for the Reporting Database

Before connecting an external reporting tool to the database, create a user with read-only permissions. Use the uyuni-setup-reportdb-user command for this:

```
usage: uyuni-setup-reportdb-user [options]

options:
  --help
            show this help message and exit
  --non-interactive
            Switches to non-interactive mode
  --dbuser=DBUSER
            Report DB User
  --dbpassword=DBPASSWORD
            Report DB Password
  --add
            Add the new user
  --delete
            Delete the user
  --modify
            Set a new password
```

21.4.2. Database Schema

The schema exports the most important tables from the main Uyuni database as a de-normalized variant containing only data relevant for reporting.

Ready-to-use reports are provided as views, aggregating data over multiple tables.

Every table has an extra column (mgm_id) specifying the Uyuni server which provided the data. On a single Uyuni server this column has the standard value 1, representing localhost. On the Uyuni Hub it is replaced with the real server ID the managed server has in the hub database.

Another common additional field is synced_date, which represents the time when the data were exported from the main Uyuni server database.

Navigate to **Help › Report Database Schema** to find the schema documentation.

21.4.3. Data Generation and Update

Uyuni uses two taskomatic jobs to manage reporting data:

update-reporting-default

Generates and updates the local reporting database from the main Uyuni database. Runs by default daily at midnight on every server.

update-reporting-hub-default

Fetches reporting data from all registered peripheral servers and aggregates it into the hub reporting database. This schedule is visible on all servers, but only performs meaningful work on a Uyuni Hub. Runs by default daily at 1:30 AM.

All times are in the local timezone of the server.

For more information about hub aggregation and schedule alignment across timezones, see **Specialized Guides › Large Deployments › Hub Reporting**.

21.4.4. Tuning

21.4.4.1. report_db_batch_size

Description	The maximum number of rows fetched and written from one database to the other in one shot.
Tune when	Out of memory errors or on processing speed problems.
Value default	2000
Value recommendation	500 - 5000

Location	/etc/rhn/rhn.conf
Example	report_db_batch_size = 4000
After changing	Monitor memory usage closely before and after the change.

21.4.5. Visualizing Reporting Data with Grafana

The Uyuni Grafana formula can connect directly to the reporting database and deploy pre-built dashboards for fleet management and compliance reporting.

Prerequisites:

- A Grafana instance configured using the Monitoring Formula. See **Specialized Guides › Salt › Salt Formula Monitoring**.
- A reporting database user created with uyuni-setup-reportdb-user (only required when using external reporting tools, as the Grafana formula automatically creates and configures a dedicated read-only database user). See Create a DB User for the Reporting Database.

The reporting dashboards provided are:

- **Hub Overview** (Hub servers only) — high-level overview of the Hub setup, listing peripheral servers and their fleets.
- **Fleet Overview & Security** — fleet composition, security posture, CVE analysis, patch compliance, proxy infrastructure, and virtualization overview.
- **Reports & History** — channels, packages, errata, actions, system inventory, user management, system history, and SCAP compliance.
- **Executive Overview** — dynamic statistics including system counts and package queries.

For configuration steps, see **Specialized Guides › Salt › Salt Formula Monitoring**.

Chapter 22. Security

22.1. Auditing

In Uyuni, you can keep track of your clients through a series of auditing tasks. You can check that your clients are up to date with all public security patches (CVEs), perform subscription matching, and use OpenSCAP to check for specification compliance.

In the Uyuni Web UI, navigate to Audit to perform auditing tasks.

22.1.1. CVE audits

A CVE (Common Vulnerabilities and Exposures) is a fix for a publicly known security vulnerability.



You must apply CVEs to your clients as soon as they become available.

Each CVE contains an identification number, a description of the vulnerability, and links to further information. CVE identification numbers use the form CVE-YEAR-XXXX.

In the Uyuni Web UI, navigate to **Audit › CVE Audit** to see a list of all clients and their current patch status.

By default, the patch data is updated at 23:00 every day. We recommend that before you begin a CVE audit you refresh the data to ensure you have the latest patches.

Procedure: Updating patch data

1. In the Uyuni Web UI, navigate to **Admin › Task Schedules** and select the cve-server-channels-default schedule.
2. Click **[cve-server-channels-bunch]**.
3. Click **[Single Run Schedule]** to schedule the task. Allow the task to complete before continuing with the CVE audit.

Procedure: Verifying patch status

1. In the Uyuni Web UI, navigate to **Audit › CVE Audit**.
2. To check the patch status for a particular CVE, type the CVE identifier in the CVE Number field.
3. Select the patch statuses you want to look for, or leave all statuses checked to look for all.

4. Click **[Audit Servers]** to check all systems, or click **[Audit Images]** to check all images.

For more information about the patch status icons used on this page, see **Reference Guide › Audit › Audit Cve Audit**.

For each system, the Actions column provides information about what you need to do to address vulnerabilities. If applicable, a list of candidate channels or patches is also given. You can also assign systems to a System Set for further batch processing.

You can use the Uyuni API to verify the patch status of your clients. Use the audit.listSystemsByPatchStatus API method. For more information about this method, see the Uyuni API Guide.

22.1.2. OVAL

The CVE Audit operation relies on two primary data sources: channels and OVAL (Open Vulnerability and Assessment Language). These two sources provide the metadata for conducting CVE audits, each serving a distinct purpose.

Channels

Channels include the updated software packages, including the patches, and provide insights into the essential patches required to address vulnerabilities.

OVAL

In contrast, OVAL data supply the information about vulnerabilities themselves, and packages that render a system vulnerable to a CVE.

While it is possible to conduct CVE audits using only channels data, synchronizing OVAL data enhances the accuracy of the results, particularly in cases involving zero-day vulnerabilities or partially patched vulnerabilities.

OVAL data is much more lightweight than channels data. For example, OVAL data for openSUSE Leap 15.4 is around 50 MB.

Having synced OVAL data only, you can already perform CVE audits and check if your systems are vulnerable or not to a CVE, but you can't apply patches since they come from channels.



Key characteristics of the OVAL feature include:

- **Enabled by default**: The feature is enabled by default and requires no additional configuration.
- **Reversible**: If needed, users can disable OVAL data and revert to the standard

channel-based CVE audit. See Disabling OVAL Data Support below.

- OVAL data is updated at 23:00 every day by default. We recommend that before you begin a CVE audit you refresh the data to ensure you have the latest vulnerabilities metadata.

The following procedures can be used to enable, disable, or update OVAL data.

Procedure: Enabling OVAL data support

1. Add or modify the following setting in file /etc/rhn/rhn.conf in the container:

```
java.cve_audit.enable_oval_metadata=true
```

2. Restart the Tomcat and Taskomatic services:

```
systemctl restart tomcat taskomatic
```

Alternatively, use the procedure for disabling OVAL data support.

Procedure: Disabling OVAL data support

1. Add or modify the following setting in rhn.conf:

```
java.cve_audit.enable_oval_metadata=false
```

2. Restart the Tomcat and Taskomatic services:

```
systemctl restart tomcat taskomatic
```

Procedure: Updating OVAL data

1. In the Uyuni Web UI, navigate to **Admin › Task Schedules** and select the oval-data-sync-default schedule.
2. Click **[oval-data-sync-bunch]**.
3. Click **[Single Run Schedule]** to schedule the task.

Allow the task to complete before continuing with the CVE audit.

22.1.2.1. Collect CPE

To be able to accurately identify what vulnerabilities apply to a certain client, we need to identify the operating system product that client uses. To do that, we collect the CPE (Common Platform Enumeration) of the client as a Salt grain, then we save it to the database.

The CPE of newly registered clients will be automatically collected and saved to the database. However, for existing clients, it is necessary to execute the Update Packages List action at least once.

Procedure: Update packages list

1. In the Uyuni Web UI, navigate to **Systems › System List › All** and select a client.
2. Then go to the Software tab and select the Packages sub-tab.
3. Click **[Update Packages List]** to update packages and collect the CPE of client.

22.1.2.2. OVAL Sources

To ensure the integrity and currency of the OVAL data, Uyuni exclusively consumes OVAL data from the official maintainers of every product. Below, you can find the list of OVAL data sources.

Table 12. OVAL Sources

Product	Source URL	OVAL supported versions in Uyuni
openSUSE Leap	https://ftp.suse.com/pub/projects/security/oval	openSUSE Leap 16 openSUSE Leap 15.6
openSUSE Leap Micro	https://ftp.suse.com/pub/projects/security/oval	openSUSE Leap Micro 6.2 openSUSE Leap Micro 6.1 openSUSE Leap Micro 6.0
SUSE Linux Enterprise Server	https://ftp.suse.com/pub/projects/security/oval	SUSE Linux Enterprise Server 16 SUSE Linux Enterprise Server 15 SUSE Linux Enterprise Server 12
SUSE Linux Enterprise Desktop	https://ftp.suse.com/pub/projects/security/oval	SUSE Linux Enterprise Desktop 15

Product	Source URL	OVAL supported versions in Uyuni
SUSE Linux Enterprise Micro	https://ftp.suse.com/pub/projects/security/oval	SUSE Linux Enterprise Micro 5.5 SUSE Linux Enterprise Micro 5.4 SUSE Linux Enterprise Micro 5.3 SUSE Linux Enterprise Micro 5.2
SUSE Linux Micro	https://ftp.suse.com/pub/projects/security/oval	SUSE Linux Micro 6.2 SUSE Linux Micro 6.1 SUSE Linux Micro 6.0
SUSE Liberty Linux	https://ftp.suse.com/pub/projects/security/oval	SUSE Liberty Linux 10 SUSE Liberty Linux 9 SUSE Liberty Linux 8 SUSE Liberty Linux 7
RedHat Enterprise Linux	https://www.redhat.com/security/data/oval/v2	RedHat Enterprise Linux 9 RedHat Enterprise Linux 8 RedHat Enterprise Linux 7
Debian	https://www.debian.org/security/oval	Debian 13 Raspberry Pi OS
Ubuntu	https://security-metadata.canonical.com/oval	Ubuntu 26.04 Ubuntu 24.04 Ubuntu 22.04
AlmaLinux	https://security.almalinux.org/oval	AlmaLinux 10 AlmaLinux 9 AlmaLinux 8
Oracle Linux	https://linux.oracle.com/security/oval/	Oracle Linux 10 Oracle Linux 9 Oracle Linux 8 Oracle Linux 7



OVAL metadata is used in CVE auditing for only a subset of clients, namely, clients that use openSUSE Leap, SUSE enterprise products, RHEL, Debian or Ubuntu. This is due to the absence of OVAL vulnerability definitions metadata for the other products.

22.1.3. CVE Status

The CVE status of clients is usually either affected, not affected, or patched. These statuses are based only on the information that is available to Uyuni.

Within Uyuni, these definitions apply:

System affected by a certain vulnerability

A system which has an installed package with version lower than the version of the same package in a relevant patch marked for the vulnerability.

System not affected by a certain vulnerability

A system which has no installed package that is also in a relevant patch marked for the vulnerability.

System patched for a certain vulnerability

A system which has an installed package with version equal to or greater than the version of the same package in a relevant patch marked for the vulnerability.

Relevant patch

A patch known by Uyuni in a relevant channel.

Relevant channel

A channel managed by Uyuni, which is either assigned to the system, the original of a cloned channel which is assigned to the system, a channel linked to a product which is installed on the system or a past or future service pack channel for the system.



Because of the definitions used within Uyuni, CVE audit results might be incorrect in some circumstances. For example, unmanaged channels, unmanaged packages, or non-compliant systems might report incorrectly.

22.2. Set up a Client to Master Validation Fingerprint

In highly secure network configurations you may wish to ensure your Salt clients are connecting a specific master. To set up validation from client to master start by entering the master's fingerprint within a Salt minion configuration file:

- /etc/salt/minion.d/custom.conf in cases of using classic Salt minion in your client, or
- /etc/venv-salt-minion/minion.d/custom.conf in case of using Salt Bundle in your client

and follow the procedure:



To access a shell inside the Server container run mgrctl term on the container host.

Procedure: Adding Master's Fingerprint to Client

1. On the master, at the command prompt, as root, use this command to find the master.pub fingerprint:

```
salt-key -F master
```

   On your client, open the /etc/salt/minion.d/custom.conf or /etc/venv-salt-minion/minion.d/custom.conf configuration file. Add this line to enter the master's fingerprint replacing the example fingerprint:

```
master_finger: 'ba:30:65:2a:d6:9e:20:4f:d8:b2:f3:a7:d4:65:11:13'
```

2. Restart the service. For salt-minion, run:

```
systemctl restart salt-minion
```

3. Or, for venv-salt-minion, run:

```
systemctl restart venv-salt-minion
```

For more information about Salt Bundle, see **Client Configuration Guide › Contact Methods Saltbundle**.

For information on configuring security from a client, see https://docs.saltproject.io/en/latest/ref/configuration/minion.html.

22.3. Mirror Source Packages

If you build your own packages locally, or if you require the source code for your packages for legal reasons, it is possible to mirror the source packages on Uyuni Server.

Mirroring source packages can consume a significant amount of disk space.

Use mgrctl term before running steps inside the server container.

Procedure: Mirroring Source Packages

1. Open the /etc/rhn/rhn.conf configuration file, and add this line:

```
server.sync_source_packages = 1
```

2. Restart the Spacewalk service to pick up the changes:

```
mgradm restart
```

Currently, this feature can only be enabled globally for all repositories. It is not possible to select individual repositories for mirroring.

When this feature has been activated, the source packages become available in the Uyuni Web UI after the next repository synchronization. They are shown as sources for the binary package, and can be downloaded directly from the Web UI. Source packages cannot be installed on clients using the Web UI.

22.4. System Security with OpenSCAP

Uyuni uses OpenSCAP to audit clients. SCAP content, tailoring files, and scan policies are managed centrally on the Uyuni server, and the server automatically deploys the required files to managed systems at scan time. This allows you to schedule and view compliance scans for any client, apply built-in or custom remediation, and set up recurring compliance monitoring.

22.4.1. About SCAP

The Security Content Automation Protocol (SCAP) is a synthesis of interoperable specifications derived from community ideas. It is a line of specifications maintained by the National Institute of Standards and Technology (NIST) for maintaining system security for enterprise systems.

SCAP was created to provide a standardized approach to maintaining system security, and the standards that are used continually change to meet the needs of the community and enterprise businesses. New specifications are governed by NIST's SCAP Release cycle to provide a consistent and repeatable revision work flow. For more information, see:

- https://csrc.nist.gov/projects/security-content-automation-protocol
- https://www.open-scap.org/features/standards/
- https://ncp.nist.gov/repository?scap

Uyuni uses OpenSCAP to implement the SCAP specifications. OpenSCAP is an auditing tool that utilizes the Extensible Configuration Checklist Description Format (XCCDF). XCCDF is a standard way of expressing checklist content and defines security checklists. It also combines with other specifications such as Common Platform Enumeration (CPE), Common Configuration Enumeration (CCE), and Open Vulnerability and Assessment Language (OVAL), to create a SCAP-expressed checklist that can be processed by SCAP-validated products.

OpenSCAP checks system security configuration settings and examines systems for signs of compromise, using rules based on standards and specifications. SUSE provides the SCAP Security Guide packages as a source of this content for SUSE systems; other operating system vendors provide equivalent content for their own distributions. For more information about the SUSE Security Team, see https://www.suse.com/support/security.

22.4.2. Prepare clients for an SCAP scan

Before you begin, you need to prepare your client systems for SCAP scanning.

OpenSCAP auditing is not available on Salt clients that use the SSH contact method.

Scanning clients can consume a lot of memory and compute power on the client being scanned. Ensure you have at least 2 GB of RAM available on each client to be scanned (especially for Red Hat-based and other modern clients).

Install the OpenSCAP scanner package on the client before you begin. Depending on the operating system, this package is included either on the base operating system, or in the Uyuni Client Tools.

You do not need to install the SCAP Security Guide (content) package on the client itself. Content files are uploaded to the Uyuni server and deployed automatically at scan time. Install the content package on any system (for example, your workstation, or temporarily on a client) to obtain the content files for upload, or obtain them directly from your operating system vendor. See **Administration Guide › SCAP content management**.

The table below lists the scanner package you need, and the content package that provides default content files for that operating system:

Table 13. OpenSCAP packages

Operating system	Version	Scanner (install on client)	Content (source of content files)
SLES	12, 15	openscap-utils	scap-security-guide
openSUSE	Leap 15, Tumbleweed	openscap-utils	scap-security-guide
RHEL / CentOS / Oracle Linux / Rocky Linux / AlmaLinux	7, 8, 9, 10	openscap-utils	scap-security-guide-redhat
SUSE Liberty [1]	7, 8, 9, 10	openscap-utils	scap-security-guide
Debian	12	openscap-scanner	scap-security-guide-debian
Debian	13	openscap-scanner	Not officially supported [2]
Ubuntu	22.04	libopenscap8	scap-security-guide-ubuntu

Operating system	Version	Scanner (install on client)	Content (source of content files)
Ubuntu	24.04	openscap-scanner	Not officially supported [3]
Ubuntu	26.04	openscap-scanner	Not officially supported [4]

For Debian 13, Ubuntu 24.04, and Ubuntu 26.04, SUSE does not provide custom security guide packages. To perform scans on these clients, you must install the upstream packages (ssg-debian and ssg-debderived respectively) from their respective community repositories. Please note that these upstream guides are not officially supported by SUSE.

Check that the content package provided by your operating system vendor is up to date. For example, the scap-security-guide package provided by RHEL 7 and compatible systems has previously been found to contain outdated content. In that case, use the scap-security-guide-* package from the Uyuni Client Tools if it is provided for your operating system, or obtain content from another reliable source, unless your vendor advises otherwise.

SUSE provides the scap-security-guide package for different openscap profiles. In the current version of scap-security-guide, SUSE supports the following profiles:

- DISA STIG profile for SUSE Linux Enterprise Server 12 and 15
- ANSSI-BP-028 profile for SUSE Linux Enterprise Server 12 and 15
- PCI-DSS profile for SUSE Linux Enterprise Server 12 and 15
- HIPAA profile for SUSE Linux Enterprise Server 15
- Hardening for Public Cloud Image of SUSE Linux Enterprise Server for SAP Applications 15
- Public Cloud Hardening for SUSE Linux Enterprise 15
- Standard System Security profile for SLE 12 and 15

For Non-SUSE operating systems the included profiles are community supplied. They are not officially supported by SUSE.

22.4.3. SCAP content management

OpenSCAP uses SCAP content files to define test rules. These content files are created based on the XCCDF or OVAL standards. To get content files for upload to the Uyuni server, you can install the SCAP Security Guide package for your operating system (see Prepare clients for an SCAP scan) on any system to extract its default content files, obtain content files directly from your operating system vendor, download content from a

reputable public source (such as the NIST National Checklist Program), or create your own custom content files if you are familiar with XCCDF or OVAL.



Custom or third-party SCAP content, including content downloaded from public repositories or created in-house, is not verified by SUSE and may fall outside your support agreement.

While running compliance scans with this content is read-only and safe for target systems, applying remediation derived from unverified sources can make unexpected system changes. Always test custom remediation in a non-production environment before applying it in production.

SCAP content files (DataStream and XCCDF) are uploaded and managed centrally on the Uyuni server. The server automatically deploys the required files to the managed system at scan time, so you do not need to manually transfer content files to clients.

22.4.3.1. Upload SCAP content

Procedure: Uploading SCAP content

1. Navigate to **Audit › OpenSCAP › SCAP Content**.
2. Click **[Upload]**.
3. Provide a **Name** and optional **Description**.
4. Upload the files:
    - **DataStream file**: The filename must end with -ds.xml.
    - **XCCDF file**: The filename must end with -xccdf.xml.
    - Both files must share the same base name (for example, ssg-sle15-ds.xml and ssg-sle15-xccdf.xml).
5. Click **[Upload]**.

Uploaded files are stored on the server at /srv/susemanager/scap/ssg/content/.

22.4.3.2. Edit and delete SCAP content

Procedure: Editing and deleting SCAP content

1. Go to **Audit › OpenSCAP › SCAP Content**.
2. Click a content entry to edit its name, description, or replace the uploaded files.
3. Click **[Delete]** to remove SCAP content.



Deleting SCAP content that is referenced by a policy will break that policy. Ensure no active policies depend on the content before deleting it.

22.4.3.3. OVAL files

Due to the large size of OVAL (Open Vulnerability and Assessment Language) files, they are not transferred automatically from the server to the managed system. If your SCAP evaluation requires OVAL files, you must ensure they are already present on the managed system before scheduling the scan.

You can specify OVAL file paths when creating a SCAP policy.

22.4.4. Find OpenSCAP profiles

When you create a SCAP policy, or schedule a one-off scan by selecting SCAP content directly, in the Web UI, the list of available profiles is loaded automatically from the selected content file, so you do not usually need to look them up manually. Use the method in this section if you need to decide which content file to upload when a package provides several, or if you need the exact profile ID to pass to the system.scap.scheduleXccdfScanCustom API method, for example.

Different operating systems make available different OpenSCAP content files and profiles. One content file may contain more than one profile.

On the system where you installed the content package, use this command on RPM-based operating systems to determine the location of the available SCAP files:

```
rpm -ql <scap-security-guide-package-name-from-table>
```

On DEB-based operating systems, use this command to determine the location of the available SCAP files:

```
dpkg -L <scap-security-guide-package-name-from-table>
```

When you have identified one SCAP content file that suits your needs, list the profiles it contains:

```
oscap info /usr/share/xml/scap/ssg/content/ssg-sle15-ds.xml
```

```
Document type: Source Data Stream
Imported: 2021-03-24T18:14:45

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-sle15-xccdf-1.2.xml
Generated: (null)
Version: 1.2
Checklists:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-xccdf-1.2.xml
                Status: draft
                Generated: 2021-03-24
                Resolved: true
                Profiles:
                        Title: CIS SUSE Linux Enterprise 15 Benchmark
                                Id: xccdf_org.ssgproject.content_profile_cis
                        Title: Standard System Security Profile for SUSE Linux Enterprise
15
                                Id: xccdf_org.ssgproject.content_profile_standard
                        Title: DISA STIG for SUSE Linux Enterprise 15
                                Id: xccdf_org.ssgproject.content_profile_stig
                Referenced check files:
                        ssg-sle15-oval.xml
                                system: http://oval.mitre.org/XMLSchema/oval-definitions-5
                        ssg-sle15-ocil.xml
                                system: http://scap.nist.gov/schema/ocil/2

https://ftp.suse.com/pub/projects/security/oval/suse.linux.enterprise.15.xml
                                system: http://oval.mitre.org/XMLSchema/oval-definitions-5
Checks:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-oval.xml
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-ocil.xml
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-cpe-oval.xml
Dictionaries:
        Ref-Id: scap_org.open-scap_cref_ssg-sle15-cpe-dictionary.xml
```

Take a note of the file paths and profiles for performing the scan. You use this information when uploading SCAP content and when selecting a profile for a SCAP policy.

22.4.5. Tailoring file management

SCAP tailoring files allow you to customize the behavior of an SCAP profile without modifying the original content. Uyuni provides a dedicated interface for managing tailoring files.

Tailoring files are scoped per organization. Each organization manages its own set of tailoring files independently.

22.4.5.1. Upload a tailoring file

Procedure: Uploading a tailoring file

1. Navigate to **Audit › OpenSCAP › Tailoring Files**.
2. Click **[Create]**.

3. Provide a **Name** and optional **Description**.
4. Upload the tailoring file (XML format).
5. Click **[Upload]**.

Tailoring files are stored on the server at /srv/susemanager/scap/tailoring-files/.

22.4.5.2. Editing and deleting tailoring files

Procedure: Edit and delete tailoring files

1. Navigate to **Audit › OpenSCAP › Tailoring Files** list.
2. Click a tailoring file entry to edit its name, description, or replace the file.
3. Select and click **[Delete]** to remove them.

22.4.6. SCAP policies

SCAP policies define a reusable combination of SCAP content, a specific profile, and an optional tailoring file. Policies simplify the process of scheduling consistent compliance scans across your infrastructure, and are required for recurring scans.

Creating a policy is optional for a single, one-off scan. In that case you can select the SCAP content and profile directly when scheduling the scan, instead of creating a policy first. See Schedule SCAP scans.

22.4.6.1. Creating a policy

Procedure: Creating a policy

1. Navigate to **Audit › OpenSCAP › SCAP Policies**.
2. Click **[Create]**.
3. Fill in the policy details:

Policy Name

A unique name for the policy within your organization.

Description

An optional description of the policy's purpose.

SCAP Content

Select from the uploaded SCAP content.

XCCDF Profile

Select a profile from the chosen SCAP content. The list of available profiles is loaded dynamically from the content file.

Tailoring File

(Optional) Select a tailoring file to customize the selected profile.

Tailoring Profile

(Optional) If a tailoring file is selected, choose a profile from the tailoring file.

OVAL Files

(Optional) Comma-separated list of OVAL file paths that must exist on the managed system.

Advanced Arguments

(Optional) Additional oscap command-line arguments (for example, --skip-rule <rule_id>). For the options that are passed on, see Advanced arguments.

Fetch Remote Resources

(Optional) Enable to allow oscap to download remote XCCDF resources during evaluation.

4. Click **[Create]**.

22.4.6.2. Advanced arguments

Uyuni does not pass the **Advanced Arguments** field to oscap as it is. It reads the options it supports from the field, and builds the oscap command line on the managed system from them. This applies to policies, to one-off scans, and to the advancedArgs key of the API.

These options are passed on:

Option	Effect
--rule <rule_id>	Evaluate only the given rule. Can be given more than once.
--skip-rule <rule_id>	Do not evaluate the given rule. Can be given more than once.
--reference <name>:<id>	Evaluate only the rules that carry the given reference, for example stigid:RHEL-09-211010.
--cpe <file>	Use the given CPE dictionary or language for the applicability checks.
--local-files <directory>	Use the local copies of remote resources from the given directory instead of downloading them.
--datastream-id <id>	Use the given data stream of the collection.
--xccdf-id <id>	Use the given XCCDF component of the data stream.
--benchmark-id <id>	Evaluate the given benchmark, when the data stream holds more than one.
--tailoring-id <id>	Use the given tailoring component of the data stream.
--fetch-remote-resources	Download the remote content that the SCAP content refers to.
--remediate	Run the fixes of the failed rules after the evaluation.

The profile, the tailoring file, and the tailoring profile are selected with their own fields. Other options are ignored, and the scan sets the result and report options itself.

For example, to skip two rules of the selected profile:

```
--skip-rule xccdf_org.ssgproject.content_rule_package_telnet-server_removed --skip-rule
xccdf_org.ssgproject.content_rule_package_tftp-server_removed
```



The paths given to --cpe and --local-files are read on the managed system. Only the SCAP content and the tailoring file are transferred from the server.



--skip-rule, --reference, --cpe, --local-files, --datastream-id, --xccdf-id, --benchmark-id, and a repeated --rule need an updated Salt bundle (venv-salt-minion) on the managed system. With an older Salt bundle, these options are ignored, and a repeated --rule makes the scan fail.

22.4.6.3. Policy details and scan history

Click on a policy name in the list to view its details, including:

- The associated SCAP content, profile, and tailoring file.
- A scan history showing all scans that were executed using this policy, with compliance results.

22.4.6.4. Edit and delete policies

Procedure: Editing and deleting policies

1. Navigate to **Audit › OpenSCAP › SCAP Policies** list.
2. Click the edit icon to modify a policy.
3. Click **[Delete]** to remove a policy.

22.4.7. Schedule SCAP scans

22.4.7.1. Scheduling a scan for a single system

Procedure: Scheduling a scan for a single system

1. Navigate to **Systems › System List › All**.
2. Select a system then go to **Audit › OpenSCAP › Schedule**.
3. Choose how to configure the scan:
    - **Use a SCAP Policy**: Select an existing SCAP policy.
    - **One-off scan**: Select a SCAP Content entry directly, then choose a profile from the list loaded automatically from that content (and optionally a tailoring file and profile). Use this when you do not need to reuse the configuration later.
4. Choose the scan schedule:
    - **Run now**: Execute the scan immediately.
    - **Schedule for later**: Pick a specific date and time.

5. Click **[Schedule]**.



Package openscap-utils needs to be installed on the targeted minion. For more information see Prepare clients for an SCAP scan.

22.4.7.2. Schedule scans for multiple systems (SSM)

Procedure: Scheduling scans for multiple systems (SSM)

1. Add the target systems to the System Set Manager (SSM).
2. Navigate to **Systems › System Set Manager › Audit › Schedule**.
3. Choose how to configure the scan: select an existing **SCAP Policy**, or select a **SCAP Content** entry and profile directly for a one-off scan.
4. Choose the scan schedule.
5. Click **[Schedule]**.

22.4.7.3. Recurring scans

SCAP policies can be scheduled as recurring actions, enabling automated compliance monitoring.

To set up a recurring scan, run the procedure:

Procedure: Setting up a recurring scan

1. Navigate to the recurring actions configuration for a system or system group.
2. Select **SCAP Policy Scan** as the action type.
3. Choose the SCAP policy to apply.
4. Configure the recurrence schedule (for example, daily, weekly).



Recurring scans are linked to the selected SCAP policy. If you update the policy (for example, change the profile), future recurring scans will use the updated configuration.

22.4.7.4. Scheduling a scan from the API

You can also use the Uyuni API to schedule SCAP scans, with the system.scap handler.

Procedure: Scheduling a scan from the API

1. Before you begin, ensure that the client to be scanned has Python and XML-RPC libraries installed.
2. To schedule a scan using an existing SCAP policy, use system.scap.scheduleXccdfScanWithPolicy:

```
#!/usr/bin/python3
import xmlrpc.client
client = xmlrpc.client.ServerProxy('https://server.example.com/rpc/api')
key = client.auth.login('username', 'password')
client.system.scap.scheduleXccdfScanWithPolicy(key, [<1000010001>],
<policy_id>, xmlrpc.client.DateTime())
client.auth.logout(key)
```

   In this example:

   - [<1000010001>] is the list of system IDs (sids) to scan.
   - <policy_id> is the ID of an existing SCAP policy.

3. To schedule a scan without a saved policy, use system.scap.scheduleXccdfScanCustom and pass the SCAP content and profile directly:

```
#!/usr/bin/python3
import xmlrpc.client
client = xmlrpc.client.ServerProxy('https://server.example.com/rpc/api')
key = client.auth.login('username', 'password')
params = {
    'scapContentId': <scap_content_id>,
    'xccdfProfileId': '<profile_name>',
}
client.system.scap.scheduleXccdfScanCustom(key, [<1000010001>], params,
xmlrpc.client.DateTime())
client.auth.logout(key)
```

   In this example:

   - <scap_content_id> is the ID of previously uploaded SCAP content.
   - <profile_name> is the XCCDF profile ID to use (for example, xccdf_org.ssgproject.content_profile_stig).
   - params can also include the optional keys tailoringFileId,

tailoringProfileId, ovalFiles, advancedArgs, and fetchRemoteResources.

For the options that advancedArgs passes on, see Advanced arguments.

4. Run the script on the client you want to scan, from the command prompt.

22.4.8. How scans work

When a scan is scheduled:

1. The Uyuni server transfers the required SCAP content files (DataStream and XCCDF) from the server to the managed system using Salt's file management.
2. If a tailoring file is associated with the policy, it is also transferred.
3. The oscap tool is executed on the managed system with the appropriate parameters (profile, rules, tailoring, etc.).
4. Scan results (results.xml and report.html) are collected from the managed system and stored on the server.
5. Results are available for review in the WebUI under the system's audit section.

Files are transferred to /var/cache/salt/minion/scap/ on the managed system.



OVAL files are **not** transferred automatically due to their large size. If your scan requires OVAL files, they must be present on the managed system before the scan runs.

22.4.9. Scan results

Information about the scans you have run is in the Uyuni Web UI. Navigate to **Audit › OpenSCAP › All Scans** for a table of results. For more information about the data in this table, see **Reference Guide › Audit › Openscap All Scans**.

To ensure that detailed information about scans is available, you need to enable it on the client. In the Uyuni Web UI, navigate to **Admin › Organizations** and click on the organization the client is a part of. Navigate to the Configuration tab, and check the Enable Upload of Detailed SCAP Files option. When enabled, this generates an additional HTML file on every scan, which contains extra information. The results show an extra line similar to this:

```
Detailed Results: xccdf-report.html xccdf-results.xml scap-yast2sec-oval.xml.result.xml
```

To retrieve scan information from the command line, use the spacewalk-report command:

```
spacewalk-report system-history-scap
spacewalk-report scap-scan
```

```
spacewalk-report scap-scan-results
```

You can also use the Uyuni API to view results, with the system.scap handler.

22.4.10. Removing SCAP Scan Results

Only SCAP scans that have passed the configured retention period can be deleted. You can configure the retention period in the Organization Configuration section.

Procedure: Configuring the SCAP results retention period

1. Navigate to **Home › My Organization › Configuration**.
2. Under SCAP, check Allow Deletion of SCAP Results.
3. Enter a number of days in Allow Deletion After (default is 90).

Procedure: Deleting SCAP scan results

1. Navigate to **Audit › OpenSCAP › All Scans**.
2. Select the checkbox for the scan or scans you want to delete.
3. Click **[Confirm]**.

22.4.11. Remediation

You can remediate non-compliant systems either directly from SCAP scan results in the Web UI, or by running the Bash scripts and Ansible playbooks provided in the SCAP Security Guide packages.



Remediation actions change system configuration to enforce compliance, and can disable services, change permissions, or otherwise affect how a system behaves. This applies to built-in remediation, custom remediation scripts, and the Bash scripts and Ansible playbooks described below. Test remediation on non-production systems before applying it to production systems.

22.4.11.1. Remediation from scan results

After reviewing scan results, you can apply remediation for individual rules that have failed.

Procedure: Applying remediation from scan results

1. Navigate to a completed scan's results.

2. Identify the non-compliant rule.
3. Click the remediation action to apply the fix.

Built-in remediation derived from SCAP content is always a Bash script, pre-populated from the SCAP content's built-in fix elements. When you apply it, Uyuni runs it on the managed system as a remote command, as root.

SCAP content does not provide Salt-based remediation. If you prefer to remediate with a Salt state instead, define a Custom remediation for the rule and write your own Salt state in the empty text area; unlike Bash, there is no built-in content to pre-populate it.



SCAP Security Guide packages also provide standalone Ansible playbooks for full-profile hardening. Uyuni only reads Bash-type fix elements for the per-rule remediation described here; Ansible playbooks are not applied from the Web UI and must be run manually. See Run remediation using a Bash script or Ansible playbook.

22.4.11.2. Custom remediation

If the built-in remediation from the SCAP content is insufficient or you need to tailor the fix for your environment, you can define custom remediation scripts.

Custom remediation is scoped per organization and per rule.

To save a custom remediation, follow these steps:

Procedure: Saving a custom remediation

1. Navigate to the rule's remediation view.
2. Choose the script type:
    - **Bash**: A custom shell script.
    - **Salt**: A custom Salt state definition.
3. Enter or modify the remediation script.
4. Click **[Save]**.

Custom remediation overrides the default remediation from the SCAP content for the specific rule within your organization.



Custom remediation includes an audit trail. The system tracks which user created and last

modified each custom remediation script.

22.4.11.3. Deleting custom remediation

You can delete a custom remediation to revert to the default remediation provided by the SCAP content.

Navigate to the custom remediation view and click **[Delete]** for the specific script type.

22.4.11.4. Run remediation using a Bash script or Ansible playbook

Remediation Bash scripts and Ansible playbooks are provided in the same SCAP Security Guide packages to harden the client systems, independently of the Web UI remediation workflow described above. Use this method if you want to apply a full hardening profile proactively, without first running a scan, or if you prefer not to apply remediation from the Web UI for some reason.

For example:

Listing 5. bash scripts

```
/usr/share/scap-security-guide/bash/sle15-script-cis.sh
/usr/share/scap-security-guide/bash/sle15-script-standard.sh
/usr/share/scap-security-guide/bash/sle15-script-stig.sh
```

Listing 6. Ansible playbooks

```
/usr/share/scap-security-guide/ansible/sle15-playbook-cis.yml
/usr/share/scap-security-guide/ansible/sle15-playbook-standard.yml
/usr/share/scap-security-guide/ansible/sle15-playbook-stig.yml
```

You can run them using remote commands or with Ansible, after enabling Ansible in the client system.

22.4.11.4.1. Run the Bash script on single systems as a remote command

Run the Bash script as a remote command on single systems.

1. From **System › Overview** tab, select your instance. Then in **Details › Remote Commands**, write a Bash script such as:

```
#!/bin/bash
chmod +x -R /usr/share/scap-security-guide/bash
/usr/share/scap-security-guide/bash/sle15-script-stig.sh
```

2. Click **[Schedule]**.



Folder and script names change between distribution and version. Install the content package for your operating system (see Prepare clients for an SCAP scan), then use rpm -ql <package> or dpkg -L <package> to list the exact script and playbook paths available.

22.4.11.4.2. Run the bash script using System Set Manager on multiple systems

Run the Bash script as a remote command on multiple systems at once.

1. When a system group has been created click System Groups, select Use in SSM from the table.
2. From the System Set Manager, under **Misc › Remote Command**, write a Bash script such as:

```
#!/bin/bash
chmod +x -R /usr/share/scap-security-guide/bash
/usr/share/scap-security-guide/bash/sle15-script-stig.sh
```

3. Click **[Schedule]**.

22.4.12. Navigation reference

The following menu entries are available under **Audit › OpenSCAP**:

Menu path	Description
Audit › OpenSCAP › All Scans	View all SCAP scan results.
Audit › OpenSCAP › SCAP Policies	Create and manage reusable SCAP policies.
Audit › OpenSCAP › SCAP Content	Upload and manage SCAP DataStream and XCCDF files.
Audit › OpenSCAP › Tailoring Files	Upload and manage SCAP tailoring files.
Audit › OpenSCAP › XCCDF Diff	Compare two SCAP scan results.
Audit › OpenSCAP › Advanced Search	Search across all SCAP scan results.
Systems › System Details › Audit › OpenSCAP › Schedule	Schedule a SCAP scan for a single system using the policy-based UI.
Systems › SSM › Audit › Schedule	Schedule SCAP scans for multiple systems using the policy-based UI.

22.4.13. Workflow example

The following example demonstrates a typical end-to-end workflow:

1. **Upload SCAP content**:

   Navigate to **Audit › OpenSCAP › SCAP Content** and upload a SCAP Security Guide DataStream file (for example, ssg-sle15-ds.xml and ssg-sle15-xccdf.xml).

2. **Upload a tailoring file** (optional):

   Navigate to **Audit › OpenSCAP › Tailoring Files** and upload a tailoring file if you need to customize a profile.

3. **Create a SCAP policy**:

   Navigate to **Audit › OpenSCAP › SCAP Policies** and create a policy that references the uploaded content, selects a profile (for example, CIS Benchmark Level 1), and optionally includes the tailoring file.

4. **Schedule a scan**:

   Navigate to a system's **Audit › OpenSCAP › Schedule** tab, select the policy, and schedule the scan.

5. **Review results**:

   After the scan completes, review the results under the system's **Audit › OpenSCAP › List Scans** tab.

6. **Apply remediation**:

   For any non-compliant rules, apply the built-in remediation or define custom remediation scripts tailored to your environment.

7. **Set up recurring scans**:

   Configure recurring scans using the SCAP policy to maintain ongoing compliance monitoring.

22.5. Repository Metadata

You require a custom GPG key to be able to sign repository metadata.



To access a shell inside the Server container run mgrctl term on the container host.

Procedure: Generating a custom GPG key

1. As the root user, use the gpg command to generate a new key:

```
mgrctl exec -it -- gpg --full-generate-key
```

2. At the prompts, select RSA as the key type, with a size of 2048 bits, and select an appropriate expiry date for your key. Check the details for your new key, and type y to confirm.
3. At the prompts, enter a name and email address to be associated with your key. You can also add a comment to help you identify the key, if desired. When you are happy with the user identity, type O to confirm.
4. At the prompt, enter a passphrase to protect your key.
5. The key should be automatically added to your keyring. You can check by listing the keys in your keyring:

```
mgrctl exec -- gpg --list-keys
```

6. Add the password for your keyring to the /etc/rhn/signing.conf configuration file, by opening the file in your text editor and adding this line:

```
GPGPASS="password"
```

For renewing a GPG key, see **Administration Guide › Troubleshooting › Tshoot Sync**.

You can manage metadata signing on the command line using the mgr-sign-metadata-ctl command.

Procedure: Enabling metadata signing

1. You need to know the short identifier for the key to use. You can list your available public keys in short format:

```
mgrctl exec -- gpg --keyid-format short --list-keys
...
pub   rsa4096/3E7BFE0A 2019-04-02 [SC] [expires: 2029-04-01]
      A43F9EC645ED838ED3014B035CFA51BF3E7BFE0A
uid         [ultimate] SUSE Manager
```

```
sub   rsa4096/118DE7FF 2019-04-02 [E] [expires: 2029-04-01]
```

2. Enable metadata signing with the mgr-sign-metadata-ctl command:

```
mgrctl exec -- mgr-sign-metadata-ctl enable 3E7BFE0A
OK. Found key 3E7BFE0A in keyring.
DONE. Set key 3E7BFE0A in /etc/rhn/signing.conf.
DONE. Enabled metadata signing in /etc/rhn/rhn.conf.
DONE. Exported key 3E7BFE0A to /srv/susemanager/salt/gpg/mgr-keyring.gpg.
DONE. Exported key 3E7BFE0A to /var/spacewalk/gpg/<KEY_NAME>.key.
NOTE. For the changes to become effective run:
   mgr-sign-metadata-ctl regen-metadata
```

3. You can check that your configuration is correct with this command:

```
mgrctl exec -- mgr-sign-metadata-ctl check-config
```

4. Restart the container for the configuration changes to be detected.

```
mgradm restart
```

5. Schedule metadata regeneration to replace all metadata with new signed versions.

```
mgrctl exec -- mgr-sign-metadata-ctl regen-metadata
```

You can also use the mgr-sign-metadata-ctl command to perform other tasks. Use mgr-sign-metadata-ctl --help to see the complete list.

Repository metadata signing is a global option. When it is enabled, it is enabled on all software channels on the server. This means that all clients connected to the server need to trust the new GPG key to be able to install or update packages.

Procedure: Importing GPG keys on clients

1. Deploying GPG keys to the clients works with salt states.
2. Apply the highstate with the Uyuni Web UI.

For more information about troubleshooting GPG keys, see **Administration Guide › Troubleshooting › Tshoot Sync**.

[1] SUSE Liberty Linux includes a compatible scap-security-guide package. However, the version shipped with the client tools is not compatible, so it should not be used for SCAP content.

[2] This package is community-provided and not officially supported by SUSE. Upstream ssg-debian can be used at your own risk.

[3] This package is community-provided and not officially supported by SUSE. Upstream ssg-debderived can be used at your own risk.

[4] This package is community-provided and not officially supported by SUSE. Upstream ssg-debderived can be used at your own risk.

Chapter 23. Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is a security method that restricts resources access to authorized users based on their assigned roles. In Uyuni, RBAC ensures that users can only perform actions and access resources for which they have explicit authorization, enhancing security and simplifying administration.

The core principles of RBAC include:

- **Principle of Least Privilege:** Granting only the necessary access rights for users to perform their tasks.
- **Granular Control:** Providing fine-grained control over specific functionalities.
- **Separation of Duties:** Preventing a single user from having too much control over critical processes.
- **Auditability:** Allowing for clear tracking of user actions and permissions.

23.1. Architectural scope

In Uyuni, RBAC controls only the Web UI and the Uyuni API (XML-RPC and JSON-RPC) layers. The RBAC engine controls access to these interfaces.

RBAC does not control the native authentication of the Salt API (salt-api or eauth) for external users. The Salt API is an internal service. The system permits access to the Salt API port (port 9080) only from the internal loopback interface of the Uyuni Server. Only Tomcat and Taskomatic services on the server can use the Salt API. External users cannot log in to the Salt API directly. Instead, Uyuni authenticates the user, checks their RBAC permissions, and then runs Salt commands on behalf of the user. For more information about native Salt authentication and command-line access controls, see **Specialized Guides › Salt › Salt Access Control**.

23.2. Key RBAC concepts

Understanding the following core concepts is crucial for effective RBAC management:

- **Role:** A collection of permissions defining a specific set of capabilities within Uyuni.

  Roles are assigned to users, granting the user aggregated permissions.

- **Permission:** An atomic authorization to perform a specific action, access a specific web page or call a specific API endpoint within Uyuni. In Uyuni, permissions are represented by namespaces and their access modes.
- **User:** An individual account that interacts with Uyuni. Users are assigned one or more roles.
- **Namespace:** A granular unit of access control organized in a tree-like structure. Most namespaces have distinct "View" or "Modify" modes.

23.3. User roles in Uyuni

Uyuni provides predefined roles and allows for the definition of additional custom roles, optionally inheriting from a combination of other roles.

23.3.1. Predefined roles

Refer to **Administration Guide › Users › Administrator Roles** for a complete list of predefined roles and their descriptions.

23.3.2. Defining additional roles

To define additional roles, you can:

- Select a number of existing roles to inherit permissions from.
- Specify additional namespaces to grant access to.

23.4. Namespaces for fine-grained access

Namespaces provide fine-grained access control, organized in a tree-like structure. For most namespaces, access within a namespace is further granularized by "View" and "Modify" modes.

Table 14. Example: Image Management namespaces and access modes

Namespace	Access Mode	Description
cm.build	Modify	Build container or Kiwi images
cm.image.import	Modify	Import container images from a registered image store
cm.image.list	View	List all images
cm.image.list	Modify	Delete images
cm.image.overview	View	View image details, patches, packages, build log and cluster information
cm.image.overview	Modify	Inspect, rebuild, delete images
cm.profile.details	View	View details of an image profile
cm.profile.details	Modify	Create image profiles, edit profile details
cm.profile.list	View	List all image profiles
cm.profile.list	Modify	Delete image profiles

Namespace	Access Mode	Description
cm.store.details	View	View details of an image store
cm.store.details	Modify	Create image stores, edit store details
cm.store.list	View	List all image stores
cm.store.list	Modify	Delete image stores

Table 15. Example: Salt namespaces and access modes

Namespace	Access Mode	Description
salt.remote_commands	Modify	Execute remote commands on systems through the Web UI (**Salt › Remote Commands**)
salt.keys	View	View Salt client keys (**Salt › Keys**)
salt.keys	Modify	Accept, reject, or delete Salt client keys
salt.formulas	View	View formula catalog and formula configuration data (**Salt › Formula Catalog**)
api.ansible.fetch_playbook_contents	Modify	Fetch playbook content from the control node using a synchronous Salt call
api.configchannel.sync_salt_files_on_disk	Modify	Synchronize all files on disk to the current state of the database
api.proxy.bootstrap_proxy	Modify	Deploy a proxy container on a given Salt client
api.saltkey.accept	Modify	Accept a Salt client key through the API
api.saltkey.accepted_list	View	List accepted Salt keys through the API
api.saltkey.delete	Modify	Delete a Salt client key through the API
api.saltkey.denied_list	View	List denied Salt keys through the API
api.saltkey.pending_list	View	List pending Salt keys through the API
api.saltkey.reject	Modify	Reject a Salt client key through the API

Namespace	Access Mode	Description
api.saltkey.rejected_list	View	List rejected Salt keys through the API
api.system.bootstrap	Modify	Bootstrap a client system for Salt or Salt SSH management
api.system.bootstrap_with_private_ssh_key	Modify	Bootstrap a system for Salt or Salt SSH management using a private SSH key
api.system.get_minion_id_map	View	Retrieve a mapping between Salt minion identifiers and system identifiers

A comprehensive list of namespaces and their descriptions can be retrieved by making a call to the access.listNamespaces API method. Refer to Uyuni API documentation for detailed information, including request and response formats.

23.5. Managing RBAC

Managing RBAC roles and permissions is possible either through the API or through the Web UI.

23.5.1. Managing RBAC via Web UI

In the Web UI, you can manage custom RBAC roles and assign permissions by navigating to **Admin › Access Control**. From there, you can click **[Create Access Group]** to define a new role and configure its permissions. For step-by-step instructions on creating custom roles, see **Administration Guide › Users › Creating Additional Roles**. For more information about configuring access groups and permission filters in the Web UI, see **Reference Guide › Admin › Access Control**.



Predefined roles cannot be seen or managed via the Web UI. Only custom roles created within your organization are visible in the **Admin › Access Control** section.

23.5.2. Managing RBAC via API

The Uyuni API provides methods for programmaticaly managing roles, permissions and user assignments.

23.5.2.1. The access API

These API methods manage roles and their associated access:

- listNamespaces: Lists available namespaces, access modes and their descriptions in Uyuni.

- listPermissions: Lists permitted namespaces of a role.
- listRoles: Lists existing roles in Uyuni.
- createRole: Creates a new role, optionally copying permissions from existing roles.
- deleteRole: Deletes a role.
- grantAccess: Grants access to namespaces.
- revokeAccess: Revokes access to namespaces.

23.5.2.2. The user API

These API methods manage user-role assignments:

- listPermissions: Lists effective permissions of a user.
- listRoles: Lists a user's assigned roles.
- addRole: Assigns a role to a user.
- removeRole: Removes a role from a user.

For detailed API documentation, including request and response formats, refer to Uyuni API reference.

23.6. RBAC best practices

Adhering to these best practices will help maintain a secure, efficient, and manageable RBAC environment:

- **Principle of least privilege:** Always grant users the minimum permissions necessary to perform their duties. Avoid overly broad permissions.
- **Regular review:** Periodically review assigned roles and permissions for users to ensure they are still appropriate and comply with current security policies.
- **Document roles:** Clearly document the purpose and permissions of each custom role you create.
- **Separate duties:** Implement roles that enforce separation of duties to prevent a single user from having too much control over critical processes.

Chapter 24. SSL Certificates

Uyuni uses SSL certificates to ensure that clients are registered to the correct server.

Every client that uses SSL to register to the Uyuni Server checks that it is connecting to the right server by validating against a server certificate. This process is called an SSL handshake.

During the SSL handshake, the client checks that the hostname in the server certificate matches what it expects. The client also needs to check if the server certificate is trusted.

Certificate authorities (CAs) are certificates that are used to sign other certificates. All certificates must be signed by a certificate authority (CA) in order for them to be considered valid, and for clients to be able to successfully match against them.

In order for SSL authentication to work correctly, the client must trust the root CA. This means that the root CA must be installed on every client.

The default method of SSL authentication is for Uyuni to use self-signed certificates. In this case, Uyuni has generated all the certificates, and the root CA has signed the server certificate directly.

An alternative method is to use an intermediate CA. In this case, the root CA signs the intermediate CA. The intermediate CA can then sign any number of other intermediate CAs, and the final one signs the server certificate. This is referred to as a chained certificate.

If you are using intermediate CAs in a chained certificate, the root CA is installed on the client, and the server certificate is installed on the server. During the SSL handshake, clients must be able to verify the entire chain of intermediate certificates between the root CA and the server certificate, so they must be able to access all the intermediate certificates.

There are two main ways of achieving this. In older versions of Uyuni, by default, all the intermediate CAs are installed on the client. However, you could also configure your services on the server to provide them to the client. In this case, during the SSL handshake, the server presents the server certificate as well as all the intermediate CAs. This mechanism is used now as the default configuration.

By default, Uyuni uses a self-signed certificate without intermediate CAs. For additional security, you can arrange a third party CA to sign your certificates. Third party CAs perform checks to ensure that the information contained in the certificate is correct. They usually charge an annual fee for this service. Using a third party CA makes certificates harder to spoof, and provides additional protection for your installation. If you have certificates signed by a third party CA, you can import them to your Uyuni installation.

This manual describe the use of SSL certificates in two steps:

1. How to create a self-signed certificate with Uyuni tools

2. How to deploy a certificate on Uyuni Server or Proxy

In case the certificates are provided by a third party instance like an own or external PKI, step `1` can be skipped.

- For more information about creating self-signed certificates, see **Administration Guide › Ssl Certs Selfsigned**.
- For more information about importing certificates, see **Administration Guide › Ssl Certs Imported**.

24.1. Providing SSL certificates to the Uyuni containers

24.1.1. Podman

SSL certificates are stored as Podman secrets and assigned to respective containers.

Uyuni Server Podman SSL secrets:

- CA certificates
  - uyuni-ca
  - uyuni-db-ca
- Server certificate and key
  - uyuni-cert
  - uyuni-key
- Database certificate and key
  - uyuni-db-cert
  - uyuni-db-key

Uyuni Proxy Podman SSL secrets:

- CA certificate
  - uyuni-ca
- Proxy certificate and key
  - uyuni-proxy-cert
  - uyuni-proxy-key

24.2. Self-Signed SSL Certificates

By default, Uyuni uses a self-signed certificate. In this case, the certificate is created and signed by Uyuni. This method does not use an independent certificate authority to guarantee that the details of the certificate are correct. Third-party CAs perform checks to ensure that the information contained in the certificate is correct.

- For more information on third-party CAs, see **Administration Guide › Ssl Certs Imported**.
- For more information on replacing certificates, see **Administration Guide › Ssl Certs Imported › Ssl Certs Import Replace**.

This section covers how to create or re-create your self-signed certificates on new or existing installation.

The hostname of the SSL keys and certificates must match the fully qualified hostname of the machine you deploy them on.



CA certificates generated by older Uyuni versions can miss the critical flag on the X509v3 Basic Constraints extension. Strict certificate validators, including Python 3.13 and later, reject such CA certificates and can report Basic Constraints of CA cert not marked critical. If your self-signed CA is affected, create a new CA and new server certificates, then deploy the new root CA to all clients. For more information about checking the CA certificate, see **Administration Guide › Ssl Certs Imported › Ssl Certs Verify Ca Basic Constraints**. For more information about replacing certificates and deploying the root CA to clients, see **Administration Guide › Ssl Certs Imported › Ssl Certs Import Deploy Root Ca**.

24.2.1. Re-Create Existing Server Certificates

If your existing certificates have expired or stopped working for any reason, you can generate a new server certificate from the existing CA.

Procedure: Re-Creating an Existing Server Certificate

1. On the Uyuni container host, at the command prompt, regenerate the server certificate:

```
mgrctl exec -ti -- 'rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set
-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
--set-hostname="susemanager.example.com" --set-cname="example.com"'
```

   Ensure that the set-cname parameter is the fully qualified domain name of your Uyuni Server. You can use the set-cname parameter multiple times if you require multiple aliases.

The private key and the server certificate can be found inside the server container in the directory /root/ssl-build/susemanager/ as server.key and server.crt. The name of the last directory depends on the hostname used

with --set-hostname option.

Deploy or import the new certificate and key by updating container's host podman secrets. For more information about importing the just generated certificate, see **Administration Guide › Ssl Certs Imported › Ssl Certs Import Replace**.

24.2.2. Create a new CA and Server Certificates



Be careful when you need to replace the Root CA. It is possible to break the trust chain between the server and clients. If that happens, you need an administrative user to log in to every client and deploy the CA directly.

Procedure: Creating New Certificates

1. On the Uyuni container host, at the command prompt, move the old certificate directory to a new location:

```
mgrctl exec -- mv /root/ssl-build /root/old-ssl-build
```

2. Generate a new CA certificate:

```
mgrctl exec -ti -- 'rhn-ssl-tool --gen-ca --dir="/root/ssl-build" --set
-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-common-name="SUSE Manager CA Certificate" \
--set-email="name@example.com"'
```

3. Generate a new server certificate:

```
mgrctl exec -ti -- 'rhn-ssl-tool --gen-server --dir="/root/ssl-build" --set
-country="COUNTRY" \
--set-state="STATE" --set-city="CITY" --set-org="ORGANIZATION" \
--set-org-unit="ORGANIZATION UNIT" --set-email="name@example.com" \
--set-hostname="susemanager.example.top" --set-cname="example.com"'
```

Ensure that the set-cname parameter is the fully qualified domain name of your Uyuni Server. You can use the set-cname parameter multiple times if you require multiple aliases.

You need to generate a server certificate also for each proxy, using their hostnames and cnames.

24.3. Import SSL Certificates

This section covers how to configure SSL certificate for new Uyuni installation, and how to replace existing certificates.

Before you begin, ensure you have:

- A certificate authority (CA) SSL public certificate. If you are using a CA chain, all intermediate CAs must also be available.
- An SSL server private key
- An SSL server certificate
- An SSL database private key
- An SSL database certificate

All files must be in PEM format. All CA certificates in the chain must mark X509v3 Basic Constraints as critical and include CA:TRUE. For more information, see **Administration Guide › Ssl Certs Imported › Ssl Certs Verify Ca Basic Constraints**.

The hostname of the SSL server certificate must match the fully qualified hostname of the machine you deploy them on. You can set the hostnames in the X509v3 Subject Alternative Name section of the certificate. You can also list multiple hostnames if your environment requires it. Supported Key types are RSA and EC (Elliptic Curve).



In previous versions the database SSL certificates required reportdb and db as Subject Alternative Name. This is no longer needed.

Third-party authorities commonly use intermediate CAs to sign requested server certificates. In this case, all CAs in the chain are required to be available. The mgradm commands are taking care of ordering the certificates. Ideally, the root CA should be in its own file. The server certificate file should contain the server certificate first, followed by all intermediate CA certificates in order.

24.3.1. Verify CA Basic Constraints

Strict certificate validators reject CA certificates when the X509v3 Basic Constraints extension is not marked as critical. This includes Python 3.13 and later, which is used by SUSE Linux Enterprise Server 16 and openSUSE Leap 16.

Before importing certificates or upgrading an existing installation, verify every root and intermediate CA certificate used by Uyuni. For each CA certificate, run:

```
openssl x509 -in root-ca.pem -noout -text | grep -A1 "Basic Constraints"
```

The output must include critical and CA:TRUE:

```
X509v3 Basic Constraints: critical
  CA:TRUE
```

If a CA certificate does not include both values, replace or regenerate the affected CA certificate before using it

with Uyuni. Replacing only the server certificate does not fix a CA certificate that misses the critical flag. For self-signed certificates generated by Uyuni, see **Administration Guide › Ssl Certs Selfsigned › Ssl Certs Selfsigned Create Replace**. To replace a CA that is still trusted by the clients without interrupting their connection, see **Administration Guide › Ssl Certs Ca Migration**.

24.3.2. Import Certificates for New Installations

By default, Uyuni uses a self-signed certificate. Certificates can be imported with third-party certificates at the installation time.

Procedure: Importing certificates on a new Uyuni server

1. Deploy the Uyuni Server according to the instructions in **Installation and Upgrade Guide › Install Server**. Make sure to pass the correct files as parameters to mgradm install. The parameters are:

```
3rd Party SSL Certificate Flags:
      --ssl-ca-intermediate strings    Intermediate CA certificate path
      --ssl-ca-root string             Root CA certificate path
      --ssl-server-cert string         Server certificate path
      --ssl-server-key string          Server key path
      --ssl-db-ca-intermediate strings Intermediate CA certificate path for the
database if different from the server one
      --ssl-db-ca-root string          Root CA certificate path for the
database if different from the server one
      --ssl-db-cert string             Database certificate path
      --ssl-db-key string              Database key path
```

Intermediate CAs can either be available in the file which is specified with --ssl-ca-root, or specified as extra options with --ssl-ca-intermediate. The --ssl-ca-intermediate option can be specified multiple times.

24.3.3. Import certificates for new proxy installations

The proxy certificates are embedded in the generated configuration. In order to use a third-party certificate, it needs to be provided during the configuration.

Procedure: Importing certificates on a new Uyuni Proxy

1. Install the Uyuni Proxy according to the instructions in **Installation and Upgrade Guide › Install Proxy**.
2. Follow the prompts to complete setup.



Use the same certificate authority (CA) to sign all

> certificates for servers and proxies. Certificates signed with different CAs do not match.

24.3.4. Replace certificates

You can replace active certificates on your Uyuni installation with a new certificate. There are two cases to consider: replacing only the server or database certificate, and replacing the root CA.

Replacing the root certificate requires more time and planning to avoid disruption as all the registered proxies and systems will need to have the new CA in their database before switching to it at the server level.

When using third-party certificates signed by an intermediate CA, the intermediate CA certificates need to be appended to the server or database certificate file.

The order is important: first comes the server certificate, then the CAs from the one which signed the certificate to the one signed by the root CA. The root CA certificate should not be appended to the server certificate file.

Procedure: Replacing all existing certificates

1. The following considers that you have root-ca.pem, intermediate-ca1.pem, intermediate-ca2.pem, server.pem and server.key files. It may be different depending on the number of intermediate CAs in the server certificate signature chain.
2. Combine the intermediate CAs and server certificates. The order matters, the server must be first and the intermediate CAs in order. Do not add the root CA last into the chain as it will be passed separately to uyuni-ca and uyuni-db-ca secrets. If there is no intermediate CA, then you can use the server.pem instead of the combined file in the next steps.

```
cat server.pem intermediate-ca1.pem intermediate-ca2.pem >combined-server.pem
```

3. On the Uyuni container host, at the command prompt, rotate the certificates using mgradm ssl rotate:

```
mgradm ssl rotate \
   --ssl-ca-root $path_to_ca_certificate \
   --ssl-server-cert $path_to_combined_server_certificate \
   --ssl-server-key $path_to_server_key
```

24.3.5. Replace certificates on a Proxy

If you are using a Uyuni Proxy, you must regenerate and redeploy the Proxy's configuration after replacing the certificates on the Uyuni Server. This ensures the Proxy trusts the Server's new CA and uses its own updated server certificate.

Generating the Proxy configuration using the Web UI is not supported for replacing certificates. You must use spacecmd from the command line.

Procedure: Replacing certificates on a Uyuni Proxy

1. On your Uyuni Server container host, copy the Root CA certificate (ca.crt), the Proxy certificate (proxy.crt), and the Proxy private key (proxy.key) into the server container:

```
for f in ca.crt proxy.crt proxy.key; do
  mgrctl cp $f server:/tmp/$f
done
```

2. If your environment uses one or more intermediate CAs, also copy them into the container:

```
mgrctl cp intermediateCA.pem server:/tmp/intermediateCA.pem
```

3. Execute spacecmd proxy_container_config inside the server container to generate the new configuration archive, replacing the FQDNs and admin email:

```
mgrctl exec -ti 'spacecmd proxy_container_config -- -p 8022 pxy.example.com
srv.example.com 2048 email@example.com /tmp/ca.crt /tmp/proxy.crt
/tmp/proxy.key -o /tmp/config.tar.gz'
```

   If you copied an intermediate CA in the previous step, append the -i parameter:

```
mgrctl exec -ti 'spacecmd proxy_container_config -- -p 8022 -i
/tmp/intermediateCA.pem pxy.example.com srv.example.com 2048 email@example.com
/tmp/ca.crt /tmp/proxy.crt /tmp/proxy.key -o /tmp/config.tar.gz'
```

4. Copy the generated configuration archive from the server container to the Server host:

```
mgrctl cp server:/tmp/config.tar.gz .
```

5. Transfer the config.tar.gz archive to the Proxy host:

```
scp config.tar.gz root@pxy.example.com:/root/
```

6. On the Proxy host, stop the Proxy:

```
mgrpxy stop
```

7. Install the new configuration on the Proxy host:

```
mgrpxy install config.tar.gz
```

8. Start the Proxy:

```
mgrpxy start
```

If the Root CA was changed, it needs to get deployed to all the clients connected to Uyuni. This is ideally done in advance to minimize the disruption.

Procedure: Deploying the root CA on clients

1. In the Uyuni Web UI, navigate to **Systems › Overview**.
2. Check all your clients to add them to the system set manager.
3. Navigate to **Systems › System Set Manager › Overview**.
4. In the States field, click **[Apply]** to apply the system states.
5. In the Highstate page, click **[Apply Highstate]** to propagate the changes to the clients.

24.4. Migrate SSL Certificate Authorities

Use this procedure to replace the root certificate authority (CA) of a Uyuni server, for example to replace an older self-signed CA that does not mark the X509v3 Basic Constraints extension as critical. For more information, see **Administration Guide › Ssl Certs Imported › Ssl Certs Verify Ca Basic Constraints**.



Python 3.13 and later (and probably other SSL clients) validates certificates strictly and

> rejects any CA certificate that does not mark the X509v3 Basic Constraints extension as critical.

On Podman installations, the mgradm ssl commands help perform the rotation. They generate or import the new CA, update the server and database secrets, and restart the services.

The migration is done in two phases:

1. With mgradm ssl addca, you add the new CA to the trusted bundle so that both the old and new CA are trusted, and distribute it to all clients and proxies.
2. After all clients and proxies trust the new CA, mgradm ssl rotate switches the server to a certificate signed by the new CA and drops the old one.

Also consider any machine that is not managed by the server but still accesses it through the API or a browser, as it needs the new CA too. These machines are not detected by the trust check in ssl-certs-ca-migration.pdf, so track and verify them yourself.

> Do not switch the server certificate before all clients and proxies trust the new CA. A client that does not trust the new CA can no longer connect, and you have to deploy the CA to it manually.

The certificates are stored in Podman secrets, which the container reads at startup. The server CA bundle, certificate, and key are in uyuni-ca, uyuni-cert, and uyuni-key; the database equivalents are uyuni-db-ca, uyuni-db-cert, and uyuni-db-key. On Kubernetes, the root CAs are read from the uyuni-ca and db-ca config maps, while the certificates are read from the uyuni-cert and db-cert secrets. The mgradm ssl commands manage these secrets on Podman; on Kubernetes, you update the config maps and secrets manually as described below.

24.4.1. Add the New CA to the Trust Bundle

On Podman, add the new CA with mgradm ssl addca. The command generates a new self-signed root CA, or imports a third-party one, appends it to the uyuni-ca and uyuni-db-ca secrets without changing the served certificate, restarts the services, and prints the SHA-256 fingerprint of the new CA.

To generate a new self-signed CA, run the following command on the container host. Replace the subject values with the ones for your organization. The server's fully qualified domain name is detected automatically, or you can pass it as an argument:

```
mgradm ssl addca --ssl-password CA_PASSWORD \
  --ssl-country COUNTRY --ssl-state STATE --ssl-city CITY \
  --ssl-org ORGANIZATION --ssl-ou "ORGANIZATION UNIT"
```

If --ssl-password is omitted, the command prompts for it. To use a third-party root CA instead, pass its certificate and any intermediate certificates in place of the generation options. Repeat --ssl-ca-intermediate

for each intermediate CA:

```
mgradm ssl addca --ssl-ca-root /root/new-ca.pem \
  --ssl-ca-intermediate /root/intermediate-ca.pem
```

On Kubernetes, append the new root CA certificate to the uyuni-ca and db-ca config maps manually:

```
kubectl get configmap -n uyuni-server uyuni-ca -o jsonpath='{.data.ca\.crt}' >
combined.crt && \
echo "" >> combined.crt && \
cat ca.crt >> combined.crt && \
kubectl create configmap uyuni-ca --from-file=ca.crt=combined.crt --dry-run=client -o json
| \
kubectl patch -n uyuni-server configmap uyuni-ca --patch-file /dev/stdin  -o yaml && \
rm combined.crt
```

You also need to run a similar command for db-ca.

Once the server trusts the new CA, distribute it to every client and proxy. Deploy the new CA to all Salt clients by applying the highstate, as described in **Administration Guide › Ssl Certs Imported › Ssl Certs Import Deploy Root Ca**.

On each Uyuni Proxy, make it trust the new CA without a full redeployment. The proxy reads its trusted CA from the uyuni-ca Podman secret, so update that secret directly. After mgradm ssl addca, the server publishes the combined CA bundle, which contains both the old and new root CAs, at http://SERVER_FQDN/pub/RHN-ORG-TRUSTED-SSL-CERT. Download it on the proxy host, replace the secret, and restart the proxy:

```
curl -o /tmp/proxy-ca.pem http://SERVER_FQDN/pub/RHN-ORG-TRUSTED-SSL-CERT
podman secret create --replace uyuni-ca /tmp/proxy-ca.pem
systemctl restart uyuni-proxy-pod
```

Also make sure the new CA is deployed to any other machine that is not a reachable Salt minion but still accesses the server, such as proxies running on Kubernetes or unmanaged systems.

24.4.2. Confirm the Clients Trust the New CA

Before switching, confirm that the clients trust the new CA. On Podman, run mgradm ssl rotate --check-only to query the Salt minions and report which ones already trust the new CA and which do not, without performing the rotation:

```
mgradm ssl rotate --check-only
```

This check is best-effort: it only covers Salt minions that respond at the time of the check. Offline minions and non-minion systems such as proxies running on Kubernetes or unmanaged machines are not covered and must be verified manually.

To verify one of those manually, compare fingerprints. The mgradm ssl addca command already reported the SHA-256 fingerprint of the new CA. You can also recompute it from the new CA certificate. For a self-signed CA, read the staged certificate from the server:

```
mgrctl exec -- openssl x509 -in /root/ssl-build-rotation/RHN-ORG-TRUSTED-SSL-CERT \
  -noout -fingerprint -sha256
```

For a third-party CA, run the same command against the root CA file you passed to mgradm ssl addca.

The client trust file /etc/pki/trust/anchors/RHN-ORG-TRUSTED-SSL-CERT contains several certificates. List the fingerprints of all of them on a client and confirm that the new CA fingerprint is present:

```
while openssl x509 -noout -fingerprint -sha256; do :; done < /etc/pki/trust/anchors/RHN-
ORG-TRUSTED-SSL-CERT
```

24.4.3. Switch to the New CA



Only continue after all clients and proxies trust the new CA.

On Podman, switch to the new CA with mgradm ssl rotate. The command promotes the CA staged by mgradm ssl addca, issues a new server certificate signed by it, replaces the server and database certificates, keys, and CA secrets, and restarts the services. Before switching, it checks over Salt that the clients trust the new CA and refuses to continue if some do not, unless you pass --force. Like mgradm ssl rotate --check-only, this check only covers Salt minions that respond, so an offline minion that does not report does not block the rotation; verify such minions manually beforehand.

For a self-signed CA, run the following command. Use the fully qualified domain name of your Uyuni server, or allow it to be detected automatically, and repeat --ssl-cname for additional aliases:

```
mgradm ssl rotate --ssl-password CA_PASSWORD
```

To install a third-party server certificate signed by the new CA, provide the certificate, key, and root CA instead. The database reuses the server values unless you pass the matching --ssl-db-* options:

```
mgradm ssl rotate --ssl-ca-root /root/new-ca.pem \
  --ssl-ca-intermediate /root/intermediate-ca.pem \
  --ssl-server-cert /root/server.crt \
  --ssl-server-key /root/server.key
```



The mgradm ssl addca and mgradm ssl rotate commands are only available on Podman. On Kubernetes, replace the uyuni-cert and db-cert secrets with the new server certificate and key, update the uyuni-ca and db-ca config maps to contain only the new CA, then restart the server.

24.4.3.1. Rotate a Compromised CA Immediately

If the current CA is compromised, you cannot wait for the overlap period in which both the old and new CA are trusted. Pass `--emergency` to `mgradm ssl rotate` to generate a new CA together with a matching server certificate and drop the old CA in a single step, without the two-phase overlap:

```
mgradm ssl rotate --emergency --ssl-password CA_PASSWORD
```



An emergency rotation drops the old CA immediately, so any client or proxy that does not yet trust the new CA can no longer connect until you deploy the new CA to it manually. After an emergency rotation, distribute the new CA to all clients and proxies as described in ssl-certs-ca-migration.pdf, and update the proxies as described below.

24.4.4. Update Proxies

In ssl-certs-ca-migration.pdf, you already made each proxy trust the new CA by updating its `uyuni-ca` Podman secret. Now that the server uses a certificate signed by the new CA, every proxy also needs its own certificate signed by it. For each proxy, regenerate the proxy configuration and redeploy the proxy:

Installation and Upgrade Guide › Container Deployment › Proxy Deployment.

24.5. HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS) is a policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking.

On Uyuni, HSTS is enabled by default. If you need to disable it on the server, follow this procedure:

Procedure: Disabling HSTS on the server

1. On the server container host, as root, execute the following command to create a new configuration file with setting max-age=0:

```
mgrctl exec -- \
  echo 'Header always set Strict-Transport-Security "max-age=0;
includeSubDomains"' \
  > /etc/apache2/conf.d/zz-spacewalk-www-hsts.conf
```

2. Restart Apache with:

```
mgrctl exec -- systemctl restart apache2
```

If you need to disable it on the proxy, follow this procedure:

Procedure: Disabling HSTS on the proxy

1. On the server container host, as root, execute the following command to create a new configuration file with setting max-age=0:

```
echo 'Header always set Strict-Transport-Security "max-age=0;
includeSubDomains' \
  > /etc/uyuni/custom-httpd.conf
```

2. Run the command:

```
mgrpxy install --tuning-httpd /etc/uyuni/custom-httpd.conf config.tar.gz
```



When naming the new config file <filename>.conf, make sure it is loaded at the right time. For example, to override something defined in spacewalk-www.conf the new file needs to be alphabetically after this file. For more information about how Apache loads files, see https://httpd.apache.org/docs.



When HSTS is enabled while using the default SSL certificate generated by Uyuni or a self-signed certificate, browsers will refuse to connect with HTTPS unless the CA used to sign such certificates is trusted by the browser. If you are using the SSL certificate generated by Uyuni, you can trust it by importing the file located at http://<SERVER-HOSTNAME>/pub/RHN-ORG-TRUSTED-SSL-CERT to the browsers of all users.

Chapter 25. Subscription Matching

Your SUSE products require subscriptions, which are managed by the SUSE Customer Center (SCC). Uyuni runs a nightly report checking the subscription status of all your registered clients against your SCC account. The report gives you information about which clients consume which subscriptions, how many subscriptions you have remaining and available to use, and which clients do not have a current subscription.

Navigate to **Audit › Subscription Matching** to see the report.

The Subscriptions Report tab gives information about current and expiring subscriptions.

The Unmatched Products Report tab gives a list of clients that do not have a current subscription. This includes clients that could not be matched, or that are not currently registered with Uyuni. The report includes product names and the number of systems that remain unmatched.

The Pins tab allows you to associate individual clients to the relevant subscription. This is especially useful if the subscription manager is not automatically associating clients to subscriptions successfully.

The Messages tab shows all the messages generated by subscription matcher during the matching process. They provide information to help understand the results and to improve the matching.

You can also download the reports in .csv format, or access them from that command prompt in the /var/lib/spacewalk/subscription-matcher/ directory.

By default, the subscription matcher runs daily, at midnight. To change this, navigate to **Admin › Task Schedules** and click gatherer-matcher-default. Change the schedule as required, and click **[Update Schedule]**.

Because the report can only match current clients with current subscriptions, you might find that the matches change over time. The same client does not always match the same subscription. This can be due to new clients being registered or unregistered, or because of the addition or expiration of subscriptions.

The subscription matcher automatically attempts to reduce the number of unmatched products, limited by the terms and conditions of the subscriptions in your account. However, if you have incomplete hardware information, unknown virtual machine host assignments, or clients running in unknown public clouds, the matcher might show that you do not have enough subscriptions available. Always ensure you have complete data about your clients included in Uyuni, to help ensure accuracy.



The subscription matcher does not always match clients and subscriptions accurately. It is not intended to be a replacement for auditing.

25.1. Pin Clients to Subscriptions

If the subscription matcher is not automatically matching a particular client with the correct subscription, you can manually pin them. When you have created a pin, the subscription matcher favors matching a specific

subscription with a given system or group of systems.

However, the matcher does not always respect a pin. It depends on the subscription being available, and whether or not the subscription can be applied to the client. Additionally, pins are ignored if they result in a match that violates the terms and conditions of the subscription, or if the matcher detects a more accurate match if the pin is ignored.

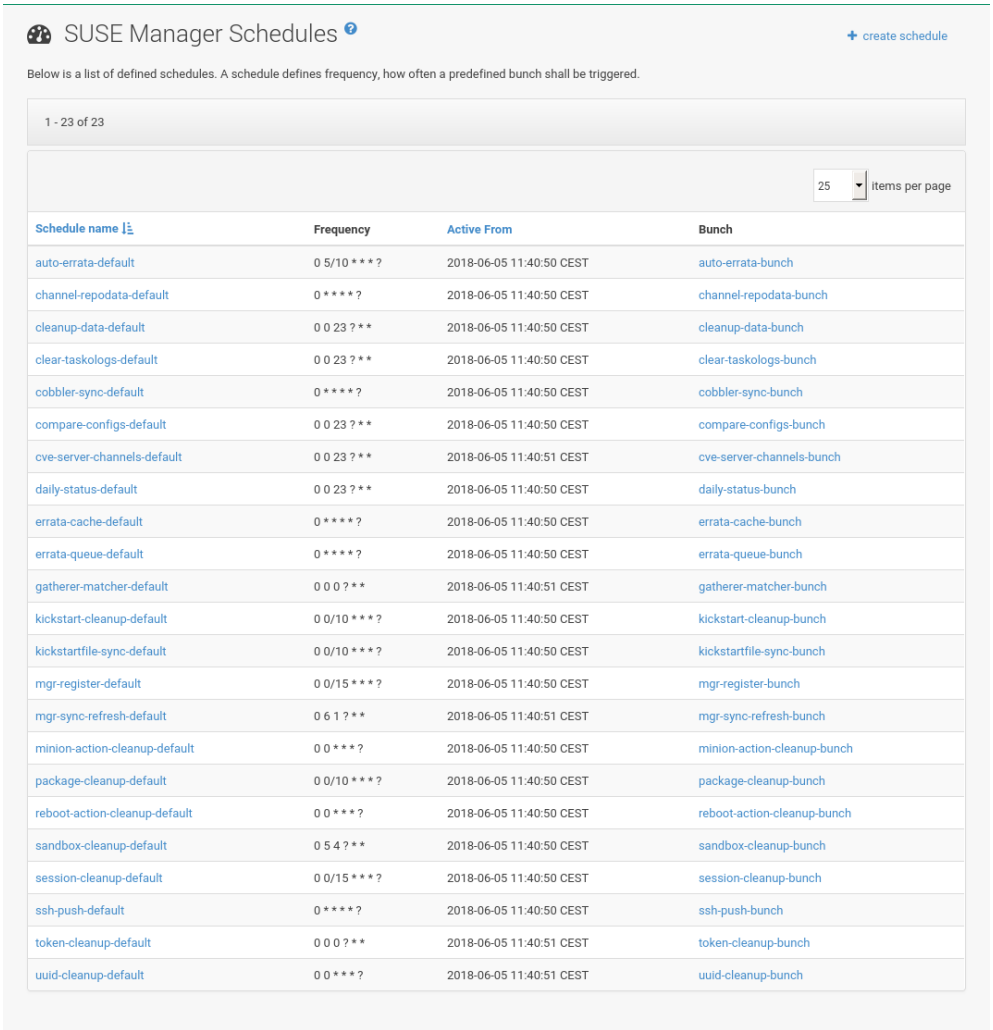
To add a new pin, click **[Add a Pin]**, and select the client to pin.



We do not recommend using pinning regularly, or for a large number of clients. The subscription matcher tool is generally accurate enough for most installations.

Chapter 26. Task Schedules

All predefined task bunches are listed under **Admin › Task Schedules**.



Schedule name	Frequency	Active From	Bunch
auto-errata-default	0 5/10 * * * ?	2018-06-05 11:40:50 CEST	auto-errata-bunch
channel-repodata-default	0 * * * * ?	2018-06-05 11:40:50 CEST	channel-repodata-bunch
cleanup-data-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	cleanup-data-bunch
clear-taskologs-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	clear-taskologs-bunch
cobbler-sync-default	0 * * * * ?	2018-06-05 11:40:50 CEST	cobbler-sync-bunch
compare-configs-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	compare-configs-bunch
cve-server-channels-default	0 0 23 ? * *	2018-06-05 11:40:51 CEST	cve-server-channels-bunch
daily-status-default	0 0 23 ? * *	2018-06-05 11:40:50 CEST	daily-status-bunch
errata-cache-default	0 * * * * ?	2018-06-05 11:40:50 CEST	errata-cache-bunch
errata-queue-default	0 * * * * ?	2018-06-05 11:40:50 CEST	errata-queue-bunch
gatherer-matcher-default	0 0 0 ? * *	2018-06-05 11:40:51 CEST	gatherer-matcher-bunch
kickstart-cleanup-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	kickstart-cleanup-bunch
kickstartfile-sync-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	kickstartfile-sync-bunch
mgr-register-default	0 0/15 * * * ?	2018-06-05 11:40:50 CEST	mgr-register-bunch
mgr-sync-refresh-default	0 6 1 ? * *	2018-06-05 11:40:51 CEST	mgr-sync-refresh-bunch
minion-action-cleanup-default	0 0 * * * ?	2018-06-05 11:40:50 CEST	minion-action-cleanup-bunch
package-cleanup-default	0 0/10 * * * ?	2018-06-05 11:40:50 CEST	package-cleanup-bunch
reboot-action-cleanup-default	0 0 * * * ?	2018-06-05 11:40:50 CEST	reboot-action-cleanup-bunch
sandbox-cleanup-default	0 5 4 ? * *	2018-06-05 11:40:50 CEST	sandbox-cleanup-bunch
session-cleanup-default	0 0/15 * * * ?	2018-06-05 11:40:50 CEST	session-cleanup-bunch
ssh-push-default	0 * * * * ?	2018-06-05 11:40:50 CEST	ssh-push-bunch
token-cleanup-default	0 0 0 ? * *	2018-06-05 11:40:51 CEST	token-cleanup-bunch
uuid-cleanup-default	0 0 * * * ?	2018-06-05 11:40:51 CEST	uuid-cleanup-bunch

Click **Uyuni Schedules › Schedule name** to open the **Schedule Name › Basic Schedule Details**. You can disable it or change its frequency.

Click **[Edit Schedule]** to update the schedule with your settings.

Click **[Disable Schedule]** in the upper right-hand corner to disable a schedule.



Only disable a schedule if you are absolutely certain this is necessary as they are essential for Uyuni to work properly.

When a task is disabled, it is still shown in the list. When you click **Uyuni Schedules › Schedule name** you can activate the job again by clicking **[Activate Schedule]**.

If you click a bunch name, a list of runs of that bunch type and their status is displayed.

Clicking the start time links takes you back to the **Schedule Name › Basic Schedule Details**.

26.1. Predefined task bunches

The following predefined task bunches are scheduled by default and can be configured:

auto-errata-default

Schedules auto errata updates as necessary.

channel-repodata-default

(Re)generates repository metadata files.

cleanup-data-default

Cleans up stale package change log and monitoring time series data from the database.

clear-taskologs-default

Clears task engine (taskomatic) history data older than a specified number of days, depending on the job type, from the database.

clm-diff-default

Compares CLM channels in a queue to their original channels and stores the differences in the reporting database. The **compare-task-default** schedule adds channels to this queue.

cobbler-sync-default

Synchronizes distribution and profile data from Uyuni to Cobbler. For more information about autoinstallation powered by Cobbler, see **Client Configuration Guide › Autoinst Intro**.

compare-configs-default

Compares configuration files as stored in configuration channels with the files stored on all configuration-enabled servers. To review comparisons, click **Systems** tab and select the system of interest. Go to **Configuration › Compare Files**. For more information, see **Reference Guide › Systems › Sd Configuration › Sd Config Compare Files**.

compare-task-default

Schedules comparison tasks, such as finding differences for CLM channels.

cve-server-channels-default

Updates internal pre-computed CVE data that is used to display results on the **Audit › CVE Audit** page. Search results in the **Audit › CVE Audit** page are updated to the last run of this schedule. For more information, see **Reference Guide › Audit › Audit Cve Audit**.

daily-status-default

Sends daily report e-mails to relevant addresses. For more information about configuring notifications for specific users, see **Reference Guide › Users › User Details**.

errata-advisory-map-sync-default

Updates internal SUSE patch vendor advisory database tables. If available, the original advisory provided by SUSE is shown in the section Vendor Advisory of each patch detail.

errata-cache-default

Updates internal patch cache database tables, which are used to look up packages that need updates for each server. Also, this sends notification emails to users that might be interested in certain patches. For more information about patches, see **Reference Guide › Patches › Patches Menu**.

errata-queue-default

Queues automatic updates (patches) for servers that are configured to receive them.

gatherer-matcher-default

Gather virtual host data by running Virtual Host Gatherer configured in Virtual Host Managers. After updated data are available, the Subscription Matcher job is run.

kickstart-cleanup-default

Cleans up stale Kickstart session data.

kickstartfile-sync-default

Generates Cobbler files corresponding to Kickstart profiles created by the configuration wizard.

mgr-forward-registration-default

Synchronizes client registration data with SUSE Customer Center. By default, new, changed, or deleted client data are forwarded. To disable synchronization set in /etc/rhn/rhn.conf, run:

```
server.susemanager.forward_registration = 0
```



Disabling data synchronizing with SCC will lead to reduced visibility of your managed clients between RMT, SMT, Uyuni and SCC-directly registered clients.

By synchronizing data, you ensure a uniform view of all registered clients.

Help us improve our services by sharing your reason for opting out

mgr-sync-refresh-default

Synchronizes with SUSE Customer Center (mgr-sync-refresh). By default, all custom channels are also synchronized as part of this task. For more information about custom channel synchronization, see

Administration Guide › Custom Channels › Custom Channel Synchronization.

minion-action-chain-cleanup-default

Cleans up outdated action chain data.

minion-action-cleanup-default

Deletes stale client action data from the file system. First it tries to complete any possibly unfinished actions by looking up the corresponding results stored in the Salt job cache. An unfinished action can occur if the server has missed the results of the action. For successfully completed actions it removes artifacts such as executed script files.

minion-checkin-default

Performs a regular check-in on clients.

notifications-cleanup-default

Cleans up expired notification messages.

oval-data-sync-default

Generate OVAL data required to increase the accuracy of CVE audit queries.

package-cleanup-default

Deletes stale package files from the file system.

reboot-action-cleanup-default

Any reboot actions pending for more than six hours are marked as failed and associated data is cleaned up from the database. For more information on scheduling reboot actions, see **Reference Guide › Systems › Sd Provisioning › Sd Power Management**.

session-cleanup-default

Cleans up stale Web interface sessions, typically data that is temporarily stored when a user logs in and then closes the browser before logging out.

ssh-service-default

Prompts clients to check in with Uyuni via SSH if they are configured with a SSH Push contact method. Also resume action chains after a reboot.

system-overview-update-queue-default

Update the systems overview data.

system-profile-refresh-default

Runs a hardware refresh on all systems. This happens only monthly and can increase load on the Uyuni

Server. The job uses **Specialized Guides › Salt › Salt Rate Limiting**. For tuning the batch size, see **Specialized Guides › Large Deployments › Tuning › Java Salt Batch Size**.

token-cleanup-default

Deletes expired repository tokens that are used by Salt clients to download packages and metadata.

update-payg-default

Collects authentication data from configured PAYG cloud instances.

update-reporting-default

Updates the local Reporting Database.

update-reporting-hub-default

Collects all reporting data from peripheral Uyuni Server and update the Hub Reporting Database.

update-system-overview-default

Regularly ensure the systems overview data are up to date.

uuid-cleanup-default

Cleans up outdated UUID records.

Chapter 27. Tuning Changelogs

Some packages have a long list of changelog entries. This data is downloaded by default, but it is not always useful information to keep. In order to limit the amount of changelog metadata which is downloaded and to save disk space, you can put a limit on how many entries to keep on disk.



Use mgrctl term before running steps inside the server container.

This configuration option is in the /etc/rhn/rhn.conf configuration file. The parameter defaults to 20. Changing this value to 0 will provide an unlimited number of entries.

```
java.max_changelog_entries = 20
```

If you set this parameter, it comes into effect only for new packages when they are synchronized.

After changing this parameter, restart services with mgradm restart.

You might like to delete and regenerate the cached data to remove older data.



Deleting and regenerating cached data can take a long time. Depending on the number of channels you have and the amount of data to be deleted, it can potentially take several hours. The task is run in the background by Taskomatic, so you can continue to use Uyuni while the operation completes, however you should expect some performance loss.

You can delete and request a regeneration of cached data from the command line:

```
spacewalk-sql -i
```

Then on the SQL database prompt, enter:

```
DELETE FROM rhnPackageRepodata;
INSERT INTO rhnRepoRegenQueue (id, CHANNEL_LABEL, REASON, FORCE)
(SELECT sequence_nextval('rhn_repo_regen_queue_id_seq'),
        C.label,
        'cached data regeneration',
        'Y'
        FROM rhnChannel C);
\q
```

Chapter 28. Users

Uyuni Administrators can add new users, grant permissions, and deactivate or delete users. If you are managing a large number of users, you can assign users to system groups to manage permissions at a group level. You can also change the system defaults for the Web UI, including language and theme defaults.



The Users menu is only available if you are logged in with the Uyuni administrator account.

To manage Uyuni users, navigate to **Users › User List › All** to see all users in your Uyuni Server. Each user in the list shows the username, real name, assigned roles, the date the user last signed in, and the current status of the user. Click **[Create User]** to create a new user account. Click the username to go to the User Details page.

To add new users to your organization, click **[Create User]**, complete the details for the new user, and click **[Create Login]**.

28.1. Password requirements

Uyuni is shipped with a selection of default values.

To ensure that all new user passwords adhere to the organization's security standards, Uyuni administrator has the option to enforce the password creation rules.

In the Web UI, navigate to **Admin › Manager Configuration › Password Policy** to define the password requirements. Use a combination of the following fields:

Min Password Length

Use this field to define the minimal length of password.

Max Password Length

Use this field to define the maximum length of password.

Require Digits

Use this field to specify whether the password must include the digits (0-9).

Require Lowercase Characters

Use this field to specify whether the password must include the lowercase characters (a-z).

Require Uppercase Characters

Use this field to specify whether the password must include the uppercase characters (A-Z).

Restrict Consecutive Characters

Use this field to specify whether the restricted consecutive characters.

Require Special Characters

Use this field to specify whether the password must include special characters.

Allowed Special Characters

This field is enable only of the Require Special Characters is selected. Use it to specify which special characters are allowed, for example !@#$%&*, and others.

Restrict Character Occurences

Use this field to specify restricted character occurences.

Max Character Occurences

Use this field to specify maximum character occurences.

Click **[Save]** to save any of the changed password settings.

Use **[Reset]** to change any settings to the default ones.



Uyuni is shipped with the following default values:

- Min Password Length: 4
- Max Password Length: 32
- Require Uppercase Characters: checked

28.2. Deactivate and delete accounts

You can deactivate or delete user accounts if they are no longer required. Deactivated user accounts can be reactivated at any time. Deleted user accounts are not visible, and cannot be retrieved.

Users can deactivate their own accounts. However, if users have an administrator role, the role must be removed before the account can be deactivated.

Deactivated users cannot log in to the Uyuni Web UI or schedule any actions. Actions scheduled by a user prior to their deactivation remain in the action queue. Deactivated users can be reactivated by Uyuni administrators.

28.3. User roles

Users can be assigned multiple roles, and there can be more than one user holding any role at any time. There must always be at least one active Uyuni Administrator.

To change a user's roles, except for the Uyuni Administrator role, navigate to **Users › User List › All**, select the user to change, and check or uncheck the administrator roles as required.

To change a user's Uyuni Administrator role, navigate to **Admin › Users** and check or uncheck Uyuni Admin? as required.

Table 16. User Role Permissions

Role Name	Description
Uyuni Administrator	Can perform all functions, including changing privileges of other users.
Organization Administrator	Manages activation keys, configurations, channels, and system groups.
Activation Key Administrator	Manages activation keys.
Image Administrator	Manages image profiles, builds, and stores.
Configuration Administrator	Manages system configuration.
Channel Administrator	Manages software channels, including making channels globally subscribable, and creating new channels.
System Group Administrator	Manages systems groups, including creating and deleting system groups, adding clients to existing groups, and managing user access to groups.
Regular User	Provides standard level of access. Newly created users are automatically assigned to this role.

28.4. Creating additional roles

With Role-Based Access Control (RBAC) in Uyuni, you can create custom roles (referred to as access groups in the Web UI) to fine-tune user permissions beyond the predefined administrator roles.

To create an access group, use the following procedure:

Procedure: Creating an access group

1. In the Web UI, navigate to **Admin › Access Control** and click **[Create Access Group]**.
2. On the Details page, fill in the following fields:
    - **Name**: Enter a unique name for the access group.

- **Description**: Enter a description explaining the purpose of the group.
- **Organization**: Select the organization to which the access group belongs.
- **Copy Permissions From**: Optionally select one or more existing access groups to inherit permissions from. Once created, the new group functions independently, and future changes to either group do not affect the other.

3. Click **[Continue]**.
4. On the Namespaces & Permissions page, select the permissions to grant to the access group:
   - Select **View** to grant read-only access to a namespace.
   - Select **Modify** to grant full access, including creating, updating, or deleting resources.
   - Use the search bar (Filter by name) to locate specific namespaces.
   - Filter permissions using the API, Web, and Only selected checkboxes.
   - Click **[Expand All]** or **[Collapse All]** to expand or collapse all namespaces in the hierarchy.
5. Click **[Continue]**.
6. On the Users page, assign users to this access group:
   - Select a user from the Search & Add Users dropdown to add them to the group.
   - Review the list of assigned users in the table (showing Username, Email, and Real Name).
   - To remove an assigned user, click the Delete icon in the Actions column.
7. Click **[Continue]**.

8. On the Review page, verify the access group details, assigned users, and granted permissions.
9. Click [Create] to finish creating the access group.

To edit an existing access group, navigate to **Admin › Access Control**, click the Edit icon in the Actions column for the group, modify settings across the tabs, and click [Update Access Group]. To delete an access group, navigate to **Admin › Access Control**, click the Delete icon in the Actions column, and confirm the deletion.

For more information about access groups and permission namespaces, see **Reference Guide › Admin › Access Control** and **Administration Guide › Role Based Access Control**.

28.5. User permissions and systems

If you have created system groups to manage your clients, you can assign groups to users for them to manage.

To assign a user to a system group, navigate to **Users › User List**, click the username to edit, and go to the System Groups tab. Check the groups to assign, and click [Update Defaults].

You can also select one or more default system groups for a user. When the user registers a new client, it is assigned to the chosen system group by default. This allows the user to immediately access the newly registered client.

To manage external groups, navigate to **Users › System Group Configuration**, and go to the External Authentication tab. Click [Create External Group] to create a new external group. Give the group a name, and assign it to the appropriate system group.

For more information about system groups, see **Reference Guide › Systems › System Groups**.

To see the individual clients a user can administer, navigate to **Users › User List**, click the username to edit, and go to the Systems tab. To carry out bulk tasks, you can select clients from the list to add them to the system set manager.

For more information about the system set manager, see **Client Configuration Guide › System Set Manager**.

28.6. Users and channel permissions

You can assign users to software channels within your organization either as a subscriber that consumes content from channels, or as an administrator, who can manage the channels themselves.

To subscribe a user to a channel, navigate to **Users › User List**, click the username to edit, and go to the **Channel Permissions › Subscription** tab. Check the channels to assign, and click [Update Permissions].

To grant a user channel management permissions, navigate to **Users › User List**, click the username to edit, and go to the **Channel Permissions › Management** tab. Check the channels to assign, and click **[Update Permissions]**.

Some channels in the list might not be subscribable. This is usually because of the users administrator status, or the channels global settings.

28.7. User default language

When you create a new user, you can choose which language to use for the Web UI. After a user has been created, you can change the language by navigating to **Home › My Preferences**.

The default language is set in the rhn.conf configuration file. To change the default language, open the /etc/rhn/rhn.conf file and add or edit this line:

```
web.locale = <LANGCODE>
```

If the parameter is not set, the default language is en_US.

These languages are available in Uyuni:

Table 17. Available Language Codes

Language code	Language	Dialect
bn_IN	Bangla	India
ca	Catalan	
de	German	
en_US	English	United States
es	Spanish	
fr	French	
gu	Gujarati	
hi	Hindi	
it	Italian	
ja	Japanese	
ko	Korean	
pa	Punjabi	

Language code	Language	Dialect
pt	Portuguese	
pt_BR	Portuguese	Brazil
ru	Russian	
ta	Tamil	
zh_CN	Chinese	Mainland, Simplified
zh_TW	Chinese	Taiwan, Traditional



Translations in Uyuni are provided by the community, and could be incorrect or incomplete. Where a translation is not available, the Web UI defaults to English (en_US).

28.7.1. User default interface theme

By default, the Uyuni Web UI uses the theme appropriate to the product you have installed. You can change the theme to reflect the Uyuni or SUSE Multi-Linux Manager colors. The SUSE Multi-Linux Manager theme also has a dark option available.

You can change the default theme in the rhn.conf configuration file. To change the default theme, open the /etc/rhn/rhn.conf file and add or edit this line:

```
web.theme_default = <THEME>
```

Table 18. Available WebUI Themes

Theme Name	Colors	Style
suse-light	SUSE Multi-Linux Manager	Light
suse-dark	SUSE Multi-Linux Manager	Dark
uyuni	Uyuni	Light

Chapter 29. Support

When you manage systems in Uyuni where you are entitled for support from SUSE, you can get the support data like supportconfig or sosreport. You will get the data from the managed client and upload it directly to SUSE Global Technical Support.

29.1. Create a service request number

Before handing over the support data to Global Technical Support, you need to generate a service request number first.

To create a service request, go to https://scc.suse.com/support/requests and follow the instructions on the screen. Write down the service request number.



Privacy statement

SUSE treats system reports as confidential data. For details about the SUSE privacy commitment, see https://www.suse.com/company/policies/privacy/.

29.2. Collect and upload support data from Uyuni to SUSE

Procedure: Collecting support data and uploading with the Uyuni Web UI

1. In the Uyuni Web UI, navigate to **Systems › System List › All** and select the client which support data should be added to the case.
2. Then navigate to the Details tab and select the Support sub-tab.
3. In the field Support Case Number enter the service request number created above.
4. In the field Upload Region select option EU or US depending on the server where you want to upload your data.
5. In the field Command-line Arguments you can enter options for tool which is used to collect the data from the target system. Which tool will be used can be found in the tip below this field.
6. In the field Earliest select the time of running this action.
7. Click button **[Schedule]**. The action is scheduled and will be executed defined time. It will collect the data from the client and upload it directly

to the upload server.



Supported Products where support data can be collected:

- Uyuni Proxy
- Uyuni Server when it is registered as Peripheral Server in a Hub Scenario
- all SUSE Linux Enterprise Server and openSUSE clients
- SUSE Liberty and compatible clients
- Ubuntu clients
- Debian clients



To upload the support data from the main Uyuni Server please still use mgradm support config from the container host.

Chapter 30. Troubleshooting

This section contains some common problems you might encounter with Uyuni, and solutions to resolving them.

30.1. Troubleshooting Autoinstallation

If you use the plain Salt (salt-minion) implementation for communication, you on your own must make sure that all the needed software (software packages) is properly available in configured client channels. It is not recommended to use this implementation together with Uyuni.

With the default Salt bundle (venv-salt-minion) implementation you only need a client tools channel as a child channel that is compatible with your client base channel. All required packages will be part of the Salt bundle.

- Check that the client tools software channel related to the base channel in your autoinstallation profile is available to your organization and your user.
- Check that the tools channel is available to your Uyuni as a child channel.
- Only if using the plain Salt (salt-minion) implementation, also check that the required packages and any dependencies are available in the associated channels.

30.2. Troubleshooting Bootstrap Repository for End-of-Life Products

When supported products are synchronized, bootstrap repositories are automatically created and regenerated on the Uyuni Server. When a product reaches end-of-life and becomes unsupported, bootstrap repositories must be created manually if you want to continue using the product.

For more information about bootstrap repositories, see **Client Configuration Guide › Bootstrap Repository**.

Procedure: Creating Bootstrap Repositories for End-Of-Life Products

1. At the command prompt of the Uyuni container host, as root, enter the server container:

```
mgrctl term
```

2. Inside the container, execute the following steps:

    a. List the available unsupported bootstrap repositories with the --force option, for example:

```
mgr-create-bootstrap-repo --list --force
1. SLE-12-SP2-x86_64
2. SLE-12-SP3-x86_64
```

b. Create the bootstrap repository by using the appropriate repository name as the product label:

```
mgr-create-bootstrap-repo --create SLE-12-SP2-x86_64 --force
```

If you do not want to create bootstrap repositories manually, you can check whether LTSS is available for the product and bootstrap repository you need.

30.3. Troubleshooting Clients Cloned Salt Clients

If you have used your hypervisor clone utility, and attempted to register the cloned Salt client, you might get this error:

```
We're sorry, but the system could not be found.
```

This is caused by the new, cloned, system having the same machine ID as an existing, registered, system. You can adjust this manually to correct the error and register the cloned system successfully.

For more information and instructions, see **Administration Guide › Troubleshooting › Tshoot Registerclones**.

30.4. Troubleshooting Container with Full Disk Event

In case a dedicated disk mounted as a persistent storage medium of a container runs out of storage space, an emergency action is needed.

Proceed as follows to solve the problem with resizing the storage medium. Run all listed commands as root on the container host.

Procedure: Resize Storage Medium

1. Increase the disk size. Actions to take depend on the installation scenario.
2. If the disk is partitioned (for example, there is a `/dev/vdb1` for disk `/dev/vdb`), run the following commands:

   Run the following commands:

   a. parted /dev/vdb

   b. (parted) print

   c. (parted) resizepart NUMBER 100% where NUMBER is the partition number shown by print command (for example, `1` if `/dev/vdb1`)

   d. (parted) quit

3. Resize the filesystem. For example, for an XFS filesystem, run the command:

```
xfs_growfs /dev/vdb1
```

After completing the procedure, an XFS filesystem should be using all space available on the disk.

30.5. Troubleshooting Corrupt Repositories

The information in the repository metadata files can become corrupt or out of date. This can create problems with updating clients. You can fix this by removing the files and regenerating it. With an new repository data file, updates should operate as expected.

Procedure: Resolving Corrupt Repository Data

1. Remove all files from /var/cache/rhn/repodata/<channel-label>. If you do not know the channel label, you can find it in the Uyuni Web UI, by navigating to **Software › Channels › Channel Label**.
2. On the container host, from the command line, execute the following command to regenerate the file in the container:

```
mgrctl exec -ti -- spacecmd softwarechannel_regenerateyumcache <channel-label>
```

30.6. Troubleshooting Custom Channel with Conflicting Packages

When setting up a custom channel with conflicting packages features such as creating a bootstrap repository can result in undefined behavior and make registering clients fail.

For example, conflicting packages with higher version numbers could be included into the bootstrap repository. Such packages (for example, python3-zmq or zeromq) may corrupt the creation of the bootstrap repository or cause issues during bootstrap of the client.

When the custom channel (for example, an EPEL channel) is added below the parent vendor channel, issues with conflicting packages cannot be solved directly. The way how to solve this is to separate the custom channel from the vendor channel. The custom channel needs to be created in a separate tree. In case that the custom channel needs to be delivered as a child, such environment can be created using Content Lifecycle Management (CLM). Sources in a CLM project can be added there from different trees. Using such an approach, the custom channel stays below the parent within the built environment. However, the vendor channel tree stays without the custom channel and the bootstrap repository. Then registering clients works correctly.

When the custom channel with the conflicting packages (salt, zeromq, and so on) is created as a child channel, following steps can help to avoid the issue:

Procedure: Avoiding Conflicting Packages in Custom Channels

1. Remove the custom channel as a child channel from parent channel. For more information, see **Administration Guide › Custom Channels › Manage Custom Channels**.
2. Create the custom channel in a separate tree. For more information, see **Administration Guide › Custom Channels › Creating Custom Channels and Repositories**.
3. To get the custom channel as a child channel within Content Lifecycle Management (CLM):
    - In the Uyuni Web UI, navigate to **Content Lifecycle**, and click **[Create Project]**. Enter Name and Label.
    - Attach Source to the project. Use the needed vendor channels and the custom channel. (sharing example using CentOS8)
    - Add environment into the Project. For example, use CentOS8.
    - To build the environment click the **[Build]** button. This creates an environment with vendor and custom channels that can be associated with an activation key and used to bootstrap the client.
4. Important note: In the CLM project, it is recommend to add a filter, which excludes the problematic or conflicting packages. Otherwise the conflicting packages with higher version numbers would be installed during the update of the client. For more information about filtering, see **Administration Guide › Content Lifecycle Examples › Exclude Higher Kernel Version**.
5. To get the latest patches into the CLM environment (with vendor and custom channel), click the **[Build]** button in the project. This is needed to re-built the environment.
    - For more information about CLM, see **Administration Guide › Content Lifecycle**.

30.7. Troubleshooting Disabling the FQDNS grain

The FQDNS grain returns the list of all the fully qualified DNS services in the system. Collecting this information is usually a fast process, but if the DNS settings have been misconfigured, it could take a much longer time. In some cases, the client could become unresponsive, or crash.

To prevent this problem, you can disable the FQDNS grain with a Salt flag. If you disable the grain, you can use a network module to provide FQDNS services, without the risk of the client becoming unresponsive.



This only applies to older Salt clients. If you registered your Salt client recently, the FQDNS grain is disabled by default.

On the Uyuni Server, at the command prompt, use this command to disable the FQDNS grain:

```
salt '*' state.sls util.mgr_disable_fqdns_grain
```

This command restarts each client and generate Salt events that the server needs to process. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls util.mgr_disable_fqdns_grain
```

Wait for the batch command to finish executing. Do not interrupt the process with `Ctrl` + `C`.

30.8. Troubleshooting Disk Space

Running out of disk space can have a severe impact on the Uyuni database and file structure which, in most cases, is not recoverable. Uyuni monitors free space in specific directories, and has configurable alerts. For more on space management, see **Administration Guide › Space Management**.

Generally, container volumes share space with the host filesystem. When the btrfs filesystem becomes full, it can prevent you from deleting files. To resolve this, you can add a small amount of storage temporarily to the filesystem using the btrfs device add command. This will allow you to delete files to gain space for further filesystem maintenance. When done with the maintenance, you can remove the temporary storage with btrfs device delete. For more information about this topic, see https://www.suse.com/support/kb/doc/?id=000018779.

You can recover disk space by removing unused software channels.

- For instructions on how to delete vendor channels, see **Administration Guide › Channel Management**.
- For instructions on how to delete custom channels, see **Administration Guide › Custom Channels**.

You can also check how often your custom channels are synchronized. For instructions on how deal with custom channel synchronization, see **Administration Guide › Custom Channels › Custom Channel Synchronization**.

You can also recover disk space by cleaning up unused activation keys, content lifecycle projects, and client registrations. You can also remove redundant database entries:

Procedure: Resolving Redundant Database Entries

1. Use the spacewalk-data-fsck command to list any redundant database entries.
2. Use the spacewalk-data-fsck --remove command to delete them.

30.8.1. Disk Check Warnings for Missing Directories

After migrating to containerized deployments from 5.0 or when using custom disk check configurations, you might see warnings for directories that no longer exist. This commonly occurs when the DISKCHECKDIRS environment variable references /var/lib/pgsql or its subdirectories.

In containerized deployments, the PostgreSQL database runs in its own container, and the /var/lib/pgsql directory no longer exists in the server container. The database files are stored in a separate database container

volume.

If you see warnings like these in the logs or notifications:

```
DISKCHECK: Directory /var/lib/pgsql does not exist
DISKCHECK: Directory /var/lib/pgsql/data does not exist
```

This indicates that the disk check configuration includes directories that are no longer valid in containerized deployments.

30.8.1.1. Cause

Prior to containerization, customers could configure custom directories to monitor by setting the spacecheck_dirs parameter in /etc/rhn/rhn.conf. A common configuration included /var/lib/pgsql to monitor database disk space.

In containerized deployments:

- The PODMAN_EXTRA_ARGS environment variable is the recommended way to pass DISKCHECKDIRS, DISKCHECKALERT, DISKTHRESHOLD, DISKCHECKALERT_FREE_MB, and DISKTHRESHOLD_FREE_MB into the container
- The legacy spacecheck_dirs parameter in /etc/rhn/rhn.conf is only used as a fallback when the corresponding environment variable is not set
- The database runs in a separate container and has its own dedicated disk check

30.8.1.2. Fix

Remove the obsolete directory references from the DISKCHECKDIRS environment variable.

Procedure: Removing Obsolete Directory References

1. Check if you have a systemd override file with custom directory configuration:

   ```
   cat /etc/systemd/system/uyuni-server.service.d/diskcheck.conf
   ```

2. If the file contains /var/lib/pgsql or other non-existent directories, update it to use only valid directories:

   ```
   cat > /etc/systemd/system/uyuni-server.service.d/diskcheck.conf << 'EOF'
   [Service]
   Environment="PODMAN_EXTRA_ARGS=--env DISKCHECKDIRS=/var/spacewalk /var/cache /srv"
   EOF
   ```

3. Apply the configuration by restarting Uyuni:

```
mgradm restart
```

30.8.1.3. Result

The disk check warnings for missing directories will no longer appear. The server container will only monitor the directories that exist within the container.

The database container has its own separate disk check that monitors the PostgreSQL data directory (/var/lib/pgsql/data) within the database container. This check is automatically configured and does not require manual intervention.

For more information about disk space management in containerized deployments, see **Administration Guide › Space Management**.

30.9. Troubleshooting Firewalls

If you are using a firewall that blocks outgoing traffic, it can either REJECT or DROP network requests. If it is set to DROP then you might find that synchronizing with the SUSE Customer Center times out.

This occurs because the synchronization process needs to access third-party repositories that provide packages for non-SUSE clients, and not just the SUSE Customer Center. When the Uyuni Server attempts to reach these repositories to check that they are valid, the firewall drops the requests, and the synchronization continues to wait for the response until it times out.

If this occurs, the synchronization takes a long time before it fails, and your non-SUSE products are not shown in the product list.

You can fix this problem in several different ways.

The simplest method is to configure your firewall to allow access to the URLs required by non-SUSE repositories. This allows the synchronization process to reach the URLs and complete successfully.

If allowing external traffic is not possible, configure your firewall to REJECT requests from Uyuni instead of DROP. This rejects requests to third-party URLs, so that the synchronization fails early rather than times out, and the products are not shown in the list.

If you do not have configuration access to the firewall, you can consider setting up a separate firewall on the Uyuni Server instead.

30.10. Troubleshooting high sync times between Uyuni Server and Proxy over WAN connections

Depending on what changes are executed in the WebUI or via an API call to distribution or system settings, cobbler sync command may be required to transfer files from Uyuni Server to Uyuni Proxy systems. To accomplish this, Cobbler uses a list of proxies specified in /etc/cobbler/settings.

Due to its design, cobbler sync is not able to sync only the changed or recently added files.

Instead, executing cobbler sync triggers a full sync of the /srv/tftpboot directory to all specified proxies configured in /etc/cobbler/settings. It is also influenced by the latency of the WAN connection between the involved systems.

The process of syncing may take a considerable amount of time to finish according to the logs in /var/log/cobbler/.

For example, it started at:

```
Thu Jun  3 14:47:35 2021 - DEBUG | running python triggers from
/var/lib/cobbler/triggers/task/sync/pre/*
Thu Jun  3 14:47:35 2021 - DEBUG | running shell triggers from
/var/lib/cobbler/triggers/task/sync/pre/*
```

and ended at:

```
Thu Jun  3 15:18:49 2021 - DEBUG | running shell triggers from
/var/lib/cobbler/triggers/task/sync/post/*
Thu Jun  3 15:18:49 2021 - DEBUG | shell triggers finished successfully
```

The transfer amount was roughly 1.8 GB. The transfer took almost 30 minutes.

By comparison, copying a single big file of the same size as /srv/tftboot completes within several minutes.

Switching to an rsync-based approach to copy files between Uyuni Server and Proxy may help to reduce the transfer and wait times.

A script to accomplish this task is available for download at https://suse.my.salesforce.com/sfc/p/1i000000gLOd/a/1i000000ll5B/B2AmvIJN2_JsAyjTQzCVP_x5ioVgd0bYN9X9NpMugS8.

The script does not accept command line options. Before running the script, you need to manually edit it and set correctly MLMHOSTNAME, MLMIP and MLMPROXY1 variables for it to work correctly.



There is no support available for individual adjustments of the script. The script and the comments inside aim to provide an overview of the process and steps to be taken into consideration. If further help is required, contact SUSE Consulting.

The proposed approach using the script is beneficial in the following environment:

- Uyuni Proxy systems are connected via a WAN connection;
- /srv/tftboot contains a high number of files for distributions and client PXE boot files, in total several thousand files;
- Any proxy in /etc/cobbler/settings has been disabled, otherwise Uyuni will continue to sync content to the proxies.

```
#proxies:
# - "MLMproxy.MLMproxy.test"
# - "MLMproxy2.MLMproxy.test"
```

Procedure: Analyzing New Sync Speed

1. Take a dump of the TCP traffic between Uyuni and the involved systems.
    - On Uyuni Server:

      ```
      tcpdump -i ethX -s 200 host <ip-address-of-susemanagerproxy> and not ssh
      ```

    - On Uyuni Proxy:

      ```
      tcpdump -i ethX -s 200 host <ip-address-of-susemanager> and not ssh
      ```

    - This will only capture a package size of 200 which is sufficient to run an analysis.
    - Adjust ethX to the respective network interface Uyuni uses to communicate with the proxy.
    - At last, ssh communication will not be captured to reduce the number of packages even further.
2. Start a cobbler sync.
    - To force a sync, delete the Cobbler json cache file first and then issue cobbler sync:

      ```
      rm /var/lib/cobbler/pxe_cache.json
      cobbler sync
      ```

3. When {command]cobbler sync is finished, stop the TCPdumps.
4. Open the TCPdumps using Wireshark, go to Statistics > Conversations and wait for the dump to be analyzed.
5. Switch to the TCP tab. The number shown on this tab gives the total number of conversations captured between Uyuni Server and Uyuni Proxy.
6. Look for the column Duration.
    - Start by sorting in ascending order to find out the minimal amount of time it took to transfer a file.

- Continue by sorting in descending order to find out the maximum values for the big files, for example kernel and initrd transfers.



Ignore ports 4505 and 4506 as these are used for Salt communication.

Analysis of the TCPdumps showed the transfer of small files with a size of approx. 1800 bytes from Uyuni Server to Proxy took around 0.3 seconds.

While there were not many big files, the high number of smaller files resulted high number of established connections as new TCP connection is created for every single transferred file.

Therefore, knowing the minimal amount of transfer time and a number of connections needed (approx. 5000 in the example), gives an approximate estimated time for the overall transfer time: 5000 * 0.3 / 60 = 25 minutes.

30.11. Troubleshooting Inactive clients

A Taskomatic job periodically pings clients to ensure they are connected. Clients are considered inactive if they have not responded to a Taskomatic check in for 24 hours or more. To see a list of inactive clients in the Web UI, navigate to **Systems › System List › Inactive**.

Clients can become inactive for a number of reasons:

- The client is not entitled to any Uyuni service.
- The client is behind a firewall that does not allow HTTPS connections.
- The client is behind a proxy that is misconfigured.
- The client is communicating with a different Uyuni Server, or the connection has been misconfigured.
- The client is not in a network that can communicate with the Uyuni Server.
- A firewall is blocking traffic between the client and the Uyuni Server.
- Taskomatic is misconfigured.

For more information about client connections to the server, see **Client Configuration Guide › Contact Methods Intro**.

For more information about configuring ports, see **Installation and Upgrade Guide › Network Requirements › Ports**.

For more information about troubleshooting firewalls, see **Administration Guide › Troubleshooting › Tshoot Firewalls**.

30.12. Troubleshooting Inter-Server Synchronization

Inter-server synchronization uses caches to manage the ISS Master and Slaves. These caches can be prone to bugs that create invalid entries. In this case, bugs can show up even after updating to a version that resolves the bug, because the cache is still using invalid entries. If you upgrade to a new version of ISS, but are still experiencing problems, clear all the caches to ensure you no longer have old entries creating a problem.

Cache errors can lead to synchronization failing with a variety of errors, but the error message will usually report something like this:

```
consider removing satellite-sync cache at /var/cache/rhn/satsync/* and re-run satellite-
sync with same options.
```

You can resolve this by deleting the cache on the ISS Master and the ISS Slave, so that synchronization completes successfully.



To access a shell inside the Server container run mgrctl term on the container host.

Procedure: Resolving ISS Caching Errors

1. On the ISS Master, at the command prompt, as root, delete the cache file for the Master:

```
rm -rf /var/cache/rhn/xml-*
```

2. Restart the service:

```
rcapache2 restart
```

3. On the ISS Master, at the command prompt, as root, delete the cache file for the Slave:

```
rm -rf /var/cache/rhn/satsync/*
```

4. Restart the service:

```
rcapache2 restart
```

30.13. Troubleshooting libvirt events errors in minion log

In earlier versions of Uyuni (such as SUSE Multi-Linux Manager 4.3), onboarding a client with the Virtualization Host system type deployed an event listener configuration file (/etc/venv-salt-minion/minion.d/libvirt-events.conf or /etc/salt/minion.d/libvirt-events.conf). This configuration enabled the Salt libvirt_events engine to monitor virtualization events on the host. In later Uyuni versions, this engine is no longer used or supported. However, when you migrate or upgrade Uyuni from an earlier version, this configuration file can remain on

previously onboarded minions.

When the minion starts, the libvirt_events engine attempts to connect to local libvirt sockets. The minion log file (/var/log/venv-salt-minion.log for Salt Bundle, or /var/log/salt/minion for classic Salt minion) repeatedly records error messages such as:

```
[salt.loaded.int.engines.libvirt_events:300 ][ERROR   ] Operation not supported: Cannot
use direct socket mode if no URI is set. For more information see
https://libvirt.org/kbase/failed_connection_after_install.html
libvirt.libvirtError: Operation not supported: Cannot use direct socket mode if no URI is
set.
```

or:

```
[salt.loaded.int.engines.libvirt_events:300 ][ERROR   ] Failed to connect socket to
'/var/run/libvirt/virtnetworkd-sock-ro': No such file or directory
libvirt.libvirtError: Failed to connect socket to '/var/run/libvirt/virtnetworkd-sock-ro':
No such file or directory
```

To resolve this issue, delete the obsolete libvirt-events.conf configuration file from the affected minion and restart the minion service.

Procedure: Removing obsolete libvirt-events configuration on the client

1. On the affected client, remove the libvirt-events.conf file:
    - For Salt Bundle (venv-salt-minion):

      ```
      rm /etc/venv-salt-minion/minion.d/libvirt-events.conf
      ```

    - For classic Salt minion:

      ```
      rm /etc/salt/minion.d/libvirt-events.conf
      ```

2. Restart the minion service:
    - For Salt Bundle (venv-salt-minion):

      ```
      systemctl restart venv-salt-minion
      ```

    - For classic Salt minion:

      ```
      systemctl restart salt-minion
      ```

Alternatively, you can remove the configuration file and restart the minion service across multiple clients by executing Salt commands on the Uyuni Server:

Procedure: Removing obsolete libvirt-events configuration using Salt

1. Remove the file using the file.remove execution module:
    - For Salt Bundle (venv-salt-minion):

```
salt '<target>' file.remove /etc/venv-salt-minion/minion.d/libvirt-
events.conf
```

    - For classic Salt minion:

```
salt '<target>' file.remove /etc/salt/minion.d/libvirt-events.conf
```

2. Restart the minion service:
    - For Salt Bundle (venv-salt-minion):

```
salt '<target>' service.restart venv-salt-minion
```

    - For classic Salt minion:

```
salt '<target>' service.restart salt-minion
```

After you remove the configuration file and restart the minion service, the minion no longer loads the libvirt_events engine, and the error messages no longer appear in the minion log.

30.14. Troubleshooting Local Issuer Certificates

Some older bootstrap scripts create a link to the local certificate in the wrong place. This results in zypper returning an Unrecognized error about the local issuer certificate. You can ensure that the link to the local issuer certificate has been created correctly by checking the /etc/ssl/certs/ directory. If you come across this problem, you should consider updating your bootstrap scripts to ensure that zypper operates as expected.

30.15. Troubleshooting Login Timeouts

By default, the Uyuni Web UI requires users to log in again after 30 minutes. Depending on your environment, you might want to adjust the login timeout value.

To adjust the value, you need to make the change in both rhn.conf and web.xml. Ensure you set the value in

seconds in /etc/rhn/rhn.conf, and in minutes in web.xml. The two values must equal the same amount of time.

For example, to change the timeout value to one hour, set the value in rhn.conf to 3600 seconds, and the value in web.xml to 60 minutes.

Procedure: Adjusting the Web UI Login Timeout Value

1. On the container host, open a command line inside the server container:

```
mgrctl term
```

   a. Open /etc/rhn/rhn.conf and add or edit this line to include the new timeout value in seconds:

```
web.session_database_lifetime = <Timeout_Value_in_Seconds>
```

   b. Save and close the file.

   c. Open /etc/tomcat/web.xml and add or edit this line to include the new timeout value in minutes:

```
<session-timeout>Timeout_Value_in_Minutes</session-timeout>
```

   d. Save and close the file.

2. On the container host, restart the server to enforce the new configuration:

```
systemctl restart uyuni-server.service
```

30.16. Troubleshooting Mail Configuration

For secure mail communication, you can enable authentication, define username and password, and enable SSL or STARTLS in /etc/rhn/rhn.conf:

```
java.smtp_server = string (default: localhost)
java.smtp_port = integer (default: 25)
java.smtp_auth = true/false (default: false)
java.smtp_ssl = true/false (default: false)
java.smtp_starttls = true/false (default: false)
java.smtp_user = string (default: null)
java.smtp_pass = string (default: null)
```

To increase the connection timeout for SMTP server communication, the following parameters can be set in /etc/rhn/rhn.conf:

```
java.smtp_timeout = integer (default: 5000)
java.smtp_connection_timeout = integer (default: 5000)
java.smtp_write_timeout = integer (default: 5000)
```

30.17. Mass Machine_id Duplication

If you are using Uyuni to manage virtual machines, you might find it useful to create clones of your VMs. A clone is a VM that uses a primary disk that is an exact copy of an existing disk.

While cloning VMs can save you a lot of time, the duplicated identifying information on the disk can sometimes cause problems.

If you try to register two machine where one is the clone of the other, you probably want Uyuni to register them as two separate clients. However, if the machine ID in both the original client and the clone is the same, Uyuni registers both clients as one system, and the data is overwritten one the second machine is registered.

This can be resolved by changing the machine ID of the clone, so that Uyuni recognizes them as two different clients. For more information and instructions on how to change machine ID, see **Administration Guide › Troubleshooting › Tshoot Registerclones**.

However, in some situations, it is neither feasible nor practical to change the machine ID on all systems, as this could disrupt other applications that depend on the existing machine IDs. In those cases, a option is available where a fake machine ID is generated to be used in the context of Uyuni.

Procedure: Generating machine ID for Uyuni at registration time

1. Create a Bootstrap script (skip this step if you already have it) For more information, see **Client Configuration Guide › Registration Bootstrap**.
2. Open the file and change the parameter:

```
GENERATE_OWN_MACHINEID=1
```

3. Start using your bootstrap strict, and machine ID will not collide anymore.

30.18. Troubleshooting Mounting /tmp with noexec

Salt runs remote commands from /tmp on the client's file system. Therefore you must not mount /tmp with the noexec option. The other way to solve this issue is to override temporary directory path with the TMPDIR environment variable specified for the Salt service to make it pointing to the directory with no noexec option set. It is recommended to use systemd drop-in configuration file /etc/systemd/system/venv-salt-minion.service.d/10-TMPDIR.conf if Salt Bundle is used, or /etc/systemd/system/salt-minion.service.d/10-TMPDIR.conf if salt-minion is used on the client. The example of the drop-in configuration file content:

```
[Service]
Environment=TMPDIR=/var/tmp
```

30.19. Troubleshooting Mounting /var/tmp with noexec

Salt SSH is using /var/tmp to deploy Salt Bundle to and execute Salt commands on the client with the bundled Python. Therefore you must not mount /var/tmp with the noexec option. It is not possible to bootstrap the clients, which have /var/tmp mounted with noexec option, with the Web UI because the bootstrap process is using Salt SSH to reach a client.

30.20. Troubleshooting Not Enough Disk Space

Check the available disk space before you begin migration. We recommend locating /var/spacewalk and /var/lib/pgsql on separate XFS file systems.

When you are setting up a separate file system, edit /etc/fstab and remove the /var/lib/pqsql subvolume. Reboot the server to pick up the changes.

To get more information about an upgrade problem, check the migration log file. The log file is located at /var/log/rhn/migration.log on the system you are upgrading.

30.21. Troubleshooting Notifications

The default lifetime of notification messages is 30 days, after which messages are deleted from the database, regardless of read status. To change this value, add or edit this line in /etc/rhn/rhn.conf:

```
java.notifications_lifetime = 30
```

To enable or disable a notification type, add or edit a line as follows in /etc/rhn/rhn.conf:

```
java.notifications_type_disabled = OnboardingFailed,ChannelSyncFailed,\
  ChannelSyncFinished,CreateBootstrapRepoFailed,StateApplyFailed,\
  PaygAuthenticationUpdateFailed,EndOfLifePeriod,SubscriptionWarning
```

For default settings and configuration options, see the usr/share/rhn/config-defaults/rhn_java.conf template file.

30.22. Troubleshooting Package Inconsistencies

When packages on a client are locked, Uyuni Server may not be able to correctly determine the set of applicable patches. When this occurs, package updates are available in the Web UI, but do not appear on the client, and attempts to update the client fail. Check package locks and exclude lists to determine if packages are locked or excluded on the client.

On the client, check package locks and exclude lists to determine if packages are locked or excluded:

- On SUSE Linux Enterprise and openSUSE, use the zypper locks command.

30.23. Troubleshooting Passing Grains to a Start Event

Every time a Salt client starts, it passes the machine_id grain to Uyuni. Uyuni uses this grain to determine if the client is registered. This process requires a synchronous Salt call. Synchronous Salt calls block other processes, so if you have a lot of clients start at the same time, the process could create significant delays.

To overcome this problem, a new feature has been introduced in Salt to avoid making a separate synchronous Salt call.

To use this feature, you can add a configuration parameter to the client configuration, on clients that support it.

To make this process easier, you can use the mgr_start_event_grains.sls helper Salt state.



This only applies to already registered clients. If you registered your Salt client recently, this config parameter is added by default.

On the Uyuni Server, at the command prompt, use this command to enable the start_event_grains configuration helper:

```
salt '*' state.sls util.mgr_start_event_grains
```

This command adds the required configuration into the client's configuration file, and applies it when the client is restarted. If you have a large number of clients, you can execute the command in batch mode instead:

```
salt --batch-size 50 '*' state.sls mgr_start_event_grains
```

30.24. Troubleshooting Proxy Connections and FQDN

Sometimes clients connected through a proxy appear in the Web UI, but do not show that they are connected through a proxy. This can occur if you are not using the fully qualified domain name (FQDN) to connect, and the proxy is not known to Uyuni.

To correct this behavior, specify additional FQDNs based on the configuration file location used:

- When placing a configuration file in the /etc/venv-salt-minion/minion.d/ directory, specify the top-level grains block:

```
grains:
  susemanager:
    custom_fqdns:
      - name.one
      - name.two
```

- Alternatively, when using /etc/venv-salt-minion/grains directly, omit the grains: line because the filename itself acts as the parent grain name:

```
susemanager:
  custom_fqdns:
    - proxy.example.com
    - proxy-alias.example.com
```

30.25. Troubleshooting Registering Cloned Clients

If you are using Uyuni to manage virtual machines, you might find it useful to create clones of your VMs. A clone is a VM that uses a primary disk that is an exact copy of an existing disk.

While cloning VMs can save you a lot of time, the duplicated identifying information on the disk can sometimes cause problems.

If you have a client that is already registered, you create a clone of that client, and then try and register the clone, you probably want Uyuni to register them as two separate clients. However, if the machine ID in both the original client and the clone is the same, Uyuni registers both clients as one system, and the existing client data is overwritten with that of the clone.

This can be resolved by changing the machine ID of the clone, so that Uyuni recognizes them as two different clients.



Each step of this procedure is performed on the cloned client. This procedure does not manipulate the original client, which remains registered to Uyuni.

Procedure: Resolving Duplicate Machine IDs in Cloned Salt Clients

1. **Initial System Configuration**

    a. On the cloned machine, change the hostname and IP addresses. Make sure /etc/hosts contains the changes you made and the correct host entries.

2. **Resolving Duplicate Machine IDs**

    a. For distributions that support systemd:

        i. If your machines have the same machine ID, as root, delete the files on each duplicated client and re-create it:

```
rm /etc/machine-id
rm /var/lib/dbus/machine-id
```

```
rm /var/lib/zypp/AnonymousUniqueId
dbus-uuidgen --ensure
systemd-machine-id-setup
```

ii. If the cloned machine also has a folder in /var/log/journal/ it needs to be renamed accordingly to the new machine ID. If names do not match, journalctl could not retrieve any log and podman logs would not show anything.

```
mv /var/log/journal/* /var/log/journal/$(cat /etc/machine-id)
```

b. For distributions that do not support systemd:

i. As root, generate a machine ID from dbus:

```
rm /var/lib/dbus/machine-id
rm /var/lib/zypp/AnonymousUniqueId
dbus-uuidgen --ensure
```



- If you are cloning a Red Hat Enterprise Linux 8.10 server that will later be liberated to SUSE Liberty Linux, you must perform extra steps to fix the kernel configuration files.
- Red Hat Enterprise Linux uses the machine ID to generate kernel entries in /boot/loader/entries. Not performing these steps will result in a mix of old and new kernel entries after the liberation, as SUSE Liberty Linux kernels will create new entries instead of replacing the old ones.
- After changing the machine ID and before liberating, run:

```
sudo rm -rf /boot/loader/entries/
sudo for ver in $(rpm -q kernel --qf '%{VERSION}-
%{RELEASE}.%{ARCH}\n'); do echo "Reinstalling kernel
$ver..."; sudo kernel-install add $ver /lib/modules/$ver;
done
sudo grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

- For more information and example on liberating Red Hat Enterprise Linux 8.10 server, see **Common Workflows Guide › Workflow Liberate Rhel with Secureboot**.

3. **Reconfiguring Salt Clients**

   a. If your clients still have the same Salt client ID, delete the minion_id file on each client (FQDN is used when it is regenerated on client restart).

      i. For Salt Minion clients:

```
rm /etc/salt/minion_id
rm -rf /etc/salt/pki
```

      ii. For Salt Bundle clients:

```
rm /etc/venv-salt-minion/minion_id
rm -rf /etc/venv-salt-minion/pki
```

   b. Delete accepted keys from the onboarding page and the system profile from Uyuni, and restart the client with.

      i. For Salt Minion clients:

```
service salt-minion restart
```

      ii. For Salt Bundle clients:

```
service venv-salt-minion restart
```

   c. Re-register the clients. Each client now has a different /etc/machine-id and should be correctly displayed on the System Overview page.

30.26. Remote root login on SL Micro

For enhanced security, new installations of SL Micro 6.2 and later do not allow password-based remote root login anymore, which affects server and proxy container hosts running on SL Micro and managed SL Micro

clients. Also, SLE Micro 5.5 clients with password-based remote root login which will when be migrated to 6.1/6.2 will suddenly lose this access and must be newly configured. For more information, see SL Micro Release Notes 6.2 (https://documentation.suse.com/releasenotes/sle-micro/html/releasenotes_sle-micro_6.2/).

While deploying components of Uyuni such as a Uyuni Proxy, by default, password-based remote root login is required. You can enable password-based remote root login with the following steps.

Procedure: Enable SSH root login with a password on SL Micro

You can enable SSH root login in two different ways. Either of them will work; choose the one that best fits your setup.

Option A: Use the preconfigured package

1. Install the package openssh-server-config-rootlogin from the UI/API on the client.
2. Reboot the container host to activate the new configuration either from UI or from terminal

Option B: Edit the SSH configuration manually

1. Add a drop-in config /etc/ssh/sshd_config.d/permit_root.conf file, with the following content:

```
PermitRootLogin yes
```

2. Reload the SSH server configuration

```
systemctl reload sshd
```

3. If connected using SSH, before disconnecting from the server, validate the SSH server is working correctly by opening a new SSH connection.

For more information about transactional-update, see https://documentation.suse.com/sle-micro/6.2/html/Micro-transactional-updates/index.html.

30.27. Troubleshooting Registering Deleted Clients



Use mgrctl term before running steps inside the server container.

Sometimes a new registration of a deleted (unregistered) client might not be possible. To solve this issue, some Salt cache files should be deleted on the Uyuni Server (Salt master) before trying to re-register again:

```
rm /var/cache/salt/master/thin/version
rm /var/cache/salt/master/thin/thin.tgz
```

30.28. Troubleshooting Registration from Web UI fails and does not show any errors

For the initial registration from the Web UI, all Salt clients are using Salt SSH.

Because of its nature, Salt SSH clients do not report errors back to the server.

However, the Salt SSH clients store a log locally at /var/log/salt-ssh.log that can be inspected for errors.

30.29. Troubleshooting Red Hat CDN Channel and Multiple Certificates

The Red Hat content delivery network (CDN) channels sometimes provide multiple certificates, but the Uyuni Web UI can only import a single certificate. If CDN presents a certificate that is different to the one the Uyuni Web UI knows about, validation fails and permission to access the repository is denied, even though the certificate is accurate. The error message received is:

```
[error]
Repository '<repo_name>' is invalid.
<repo.pem> Valid metadata not found at specified URL
History:
 - [|] Error trying to read from '<repo.pem>'
 - Permission to access '<repo.pem>' denied.
Please check if the URIs defined for this repository are pointing to a valid repository.
Skipping repository '<repo_nam' because of the above error.
Could not refresh the repositories because of errors.
HH:MM:SS RepoMDError: Cannot access repository. Maybe repository GPG keys are not imported
```

To resolve this issue, merge all valid certificates into a single .pem file, and rebuild the certificates for use by Uyuni:

Procedure: Resolving Multiple Red Hat CDN Certificates

1. On the Red Hat client, at the command prompt, as root, gather all current certificates from /etc/pki/entitlement/ in a single rh-cert.pem file:

   ```
   cat 866705146090697087.pem 3539668047766796506.pem redhat-entitlement-authority.pem >
   rh-cert.pem
   ```

2. Gather all current keys from /etc/pki/entitlement/ in a single rh-key.pem file:

```
cat 866705146090697087-key.pem 3539668047766796506-key.pem > rh-key.pem
```

You can now import the new certificates to the Uyuni Server, using the instructions in **Client Configuration Guide › Clients Rh Cdn**.

30.30. Troubleshooting Renaming Uyuni Server

If you change the hostname of the Uyuni Server locally, your Uyuni installation ceases to work properly. This is because the changes have not been made in the database, which prevents the changes from propagating out your clients and any proxies.

30.30.1. Rename server

If you need to change the hostname of the Uyuni Server, you can do so using the mgradm server rename command. This command updates the settings in the PostgreSQL database and the internal structures of Uyuni.



If your system uses a certificate signed by an external or third-party certificate authority, the new certificate containing the new hostname must be obtained before beginning the renaming process. To replace active certificates with your external certificates, see **Administration Guide › Replace certificates**.

30.30.1.1. Server configuration

The command takes no mandatory parameter, but can take the new hostname if it is not the one from the container host.

In case any SSL certificate needs to be generated to match the new hostname, the SSL CA password needs to be provided. This is safely achieved using a configuration file

Procedure: Prepare the configuration file for the SSL CA password

1. Write a config.yaml file with content like the following:

```
ssl:
  password: "<CA PASSWORD>"
```

This file will be used in the next procedure.

Procedure: Renaming Uyuni Server

1. Change the network settings of the server on the system level locally and remotely at the DNS server. You also need to provide configuration settings

for reverse name resolution. Changing network settings is done in the same way as with renaming any other system.

2. Reboot the Uyuni Server to use the new network configuration and to ensure the hostname has changed.

3. On the container host, from the command line, execute the following command. Add -c config.yaml if you created the file to store the SSL CA password:

```
mgradm server rename
```

If the new hostname is not resolvable, the command fails.

The renaming procedure also takes place during the restart of the server container. The logs can be found by running this command:

```
mgrctl exec -ti -- journalctl -u uyuni-update-config
```

Be aware that this command triggers a refresh of the pillar data for all Salt clients when restarting the server container: the time it takes to run depends on the number of registered clients.

30.30.1.2. Directly managed clients reconfiguration

Skip this procedure if clients are managed via a Uyuni proxy.

With the following procedure, reconfigure directly managed clients to make them aware of the new hostname and IP address.

Procedure: Reconfiguring directly managed clients

1. On the every client, in the Salt client configuration file, specify the name of the new Salt master (Uyuni Server). The filename is /etc/venv-salt-minion/minion.d/susemanager.conf or, if you do not use the Salt bundle, /etc/salt-minion/minion.d/susemanager.conf:

```
master: <new_hostname>
```

2. On every client, restart Salt service. Either run:

```
systemctl restart venv-salt-minion
```

Or, if you do not use the Salt bundle, run:

```
systemctl restart salt-minion
```

30.30.1.3. ClientcConnection with applying high state

Finally, to fully propagate the hostname to the Salt client configuration, apply the high state. Applying the high state will update the hostname in the repository URLs.

30.30.2. Reconfigure Proxy

Every proxy must be reconfigured. The new server certificate and key must be copied to the proxy. For more information, see **Installation and Upgrade › Install Proxy**.



If you use PXE boot through a proxy, you must check the configuration settings of the proxy. If you use PXE boot through non-containerized Uyuni Proxy 4.3, tftpsync needs to be reconfigured.

On the container host, execute:

```
mgrctl exec -ti -- configure-tftpsync.sh
```

30.31. Troubleshooting RPC Connection Timeouts

RPC connections can sometimes time out due to slow networks or a network link going down. This results in package downloads or batch jobs hanging or taking longer than expected. You can adjust the maximum time that an RPC connection can take by editing the configuration file. While this does not resolve networking problems, it does cause a process to fail rather than hang.

Procedure: Resolving RPC connection timeouts

1. On the Uyuni Server, open the /etc/rhn/rhn.conf file and set a maximum timeout value (in seconds):

```
server.timeout =`number`
```

2. On the Uyuni Proxy, open the /etc/uyuni/proxy/config.yaml file and set a maximum timeout value (in seconds). The proxy containers need to be restarted for the change to be effective:

```
timeout: `number`
```

3. On a SUSE Linux Enterprise Server client that uses zypper, open the /etc/zypp/zypp.conf file and set a maximum timeout value (in seconds):

```
## Valid values:  [0,3600]
## Default value: 180
download.transfer_timeout = 180
```

4. On a Red Hat Enterprise Linux client that uses yum, open the /etc/yum.conf file and set a maximum timeout value (in seconds):

```
timeout =`number`
```



If you limit RPC timeouts to less than 180 seconds, you risk aborting perfectly normal operations.

30.32. Troubleshooting Salt clients shown as down and DNS settings

Even if the Salt client is running, actions such as package refresh or apply states can be marked as failed with the message:

```
Minion is down or could not be contacted.
```

In this case try rescheduling the action. If rescheduling succeeds, the cause of the problem can be a wrong DNS configuration.



To access a shell inside the Server container run mgrctl term on the container host.

When the Salt client is restarted, or in case the grains are refreshed, the client calculates its FQDN grains, and it is unresponsive until the grains are proceeded. When a scheduled action on Uyuni Server is going to be executed, Uyuni Server performs a test.ping to the client before the actual action to ensure the client is actually running and the action can be triggered.

By default, Uyuni Server waits for 5 seconds to get the response from test.ping command. If the response is not received within 5 seconds, then the action is set to fail with the message that the client is down or could not be contacted.

To correct this, fix the DNS resolution on the client, so the client does not get stuck for 5 seconds while solving its FQDN.

If this is not possible, try to increase the value for java.salt_presence_ping_timeout in the /etc/rhn/rhn.conf file on the Uyuni Server to a value higher than 4.

For example:

```
mgrctl term
vim /etc/rhn/rhn.conf
java.salt_presence_ping_timeout = 6
```

After that, run:

```
mgradm restart
```



Increasing this value will cause Uyuni Server to take longer to check if a minion is unreachable or unresponsive, causing the Uyuni Server to be slower or less responsive overall.

30.33. Troubleshooting Salt SSH known hosts

If a machine or VM is reinstalled with a new SSH key, but the same IP address is used to onboard it, there is a chance that Salt will complain about a "man-in-the-middle attack". This happens because Salt's .ssh/known_hosts file still contains the old SSH fingerprint.

To remove a host from this list, run the following command:

```
mgrctl ssh knownhost remove <hostname>
```

If the host to remove is not using the default SSH port 22, you can specify the port manually:

```
mgrctl ssh knownhost remove <hostname> <port>
```

30.34. Troubleshooting Schema Upgrade Fails

If the schema upgrade fails, the database version check and all the other spacewalk services do not start. Run mgradm start on the container host for more information and hints how to proceed.



To access a shell inside the Server container run mgrctl term on the container host.

You can also run the version check directly in the container:

```
systemctl status uyuni-check-database.service
```

or

```
journalctl -u uyuni-check-database.service
```

These commands print debug information if you do not want to run the more general mgradm command.

30.35. Troubleshooting Synchronization

Synchronization can fail for a number of reasons. To get more information about a connection problem, run this command:

```
export URLGRABBER_DEBUG=DEBUG
spacewalk-repo-sync -c <channelname> <options> > /var/log/spacewalk-repo-sync-$(date +%F-
%R).log 2>&1
```

You can also check logs created by Zypper at /var/log/zypper.log

GPG Key Mismatch

Uyuni does not automatically trust third party GPG keys. If package synchronization fails, it could be because of an untrusted GPG key. You can find out if this is the case by opening /var/log/rhn/reposync and looking for an error like this:

```
['/usr/bin/spacewalk-repo-sync', '--channel', 'sle-12-sp1-ga-desktop-
nvidia-driver-x86_64', '--type', 'yum', '--non-interactive']
RepoMDError: Cannot access repository. Maybe repository GPG keys are not imported
```

To resolve the problem, you need to import the GPG key to Uyuni. For more on importing GPG keys, see **Administration Guide › Repo Metadata**.

GPG Key Removal from spacewalk-repo-sync

When a GPG key for repository has been manually imported using spacewalk-repo-sync, and this key is no longer needed (for example if the key was compromised, or was used for testing purposes only), it can be removed from the zypper RPM database used by spacewalk-repo-sync with the following command:

```
rpm --dbpath=/var/lib/spacewalk/reposync/root/var/lib/rpm/ -e gpg-pubkey-*
```

where gpg-pubkey-* is the name of the GPG key to be removed.

Renewing GPG Key

If you want to renew a GPG key, first remove the old key, and then generate and import a new one.

Checksum Mismatch

If a checksum has failed, you might see an error like this in the /var/log/rhn/reposync/*.log log file:

```
Repo Sync Errors: (50, u'checksums did not match
326a904c2fbd7a0e20033c87fc84ebba6b24d937 vs
afd8c60d7908b2b0e2d95ad0b333920aea9892eb', 'Invalid information uploaded
to the server')
```

```
The package microcode_ctl-1.17-102.57.62.1.x86_64 which is referenced by
patch microcode_ctl-8413 was not found in the database. This patch has
been skipped.
```

You can resolve this error by running the synchronization from the command prompt with the -Y option:

```
spacewalk-repo-sync --channel <channelname> -Y
```

This option verifies the repository data before the synchronization, rather than relying on locally cached checksums.

Connection Timeout

If the download times out with the following error:

```
28, 'Operation too slow. Less than 1000 bytes/sec transferred the last 300 seconds
```

You can resolve this error by specifying reposync_timeout and reposync_minrate configuration values in /etc/rhn/rhn.conf. By default, when less than 1000 bytes per second are transferred in 300 secs, the download is aborted. You can adjust the number of bytes per second with reposync_minrate, and the number of seconds to wait with reposync_timeout.

Manually Trusting the Key During reposync

It is possible that in some cases when reposync is run, you may need to accept the GPG key manually. For example:

```
# spacewalk-repo-sync -c nvidia-compute-sle-15-x86_64-we-sp3
17:07:40 ====================================
17:07:40 | Channel: nvidia-compute-sle-15-x86_64-we-sp3
17:07:40 ====================================
17:07:40 Sync of channel started.
New repository or package signing key received:
  Repository:       nvidia-compute-sle-15-x86_64-we-sp3
  Key Fingerprint:  610C 7B14 E068 A878 070D A4E9 9CD0 A493 D42D 0685
  Key Name:         cudatools <cudatools@nvidia.com>
  Key Algorithm:    RSA 4096
  Key Created:      Thu Apr 14 16:04:01 2022
  Key Expires:      (does not expire)
  Rpm Name:         gpg-pubkey-d42d0685-62589a51
    Note: Signing data enables the recipient to verify that no modifications occurred
after the data
    were signed. Accepting data with no, wrong or unknown signature can lead to a
corrupted system
    and in extreme cases even to a system compromise.
    Note: A GPG pubkey is clearly identified by its fingerprint. Do not rely on the
key's name. If
    you are not sure whether the presented key is authentic, ask the repository
provider or check
    their web site. Many providers maintain a web page showing the fingerprints of the
GPG keys they
    are using.
Do you want to reject the key, trust temporarily, or trust always? [r/t/a/?] (r):
```

30.36. Troubleshooting Taskomatic

Repository metadata regeneration is a relatively intensive process, so Taskomatic can take several minutes to complete. Additionally, if Taskomatic crashes, repository metadata regeneration can be interrupted.



To access a shell inside the Server container run mgrctl term on the container host.

If Taskomatic is still running, or if the process has crashed, package updates can seem available in the Web UI, but do not appear on the client, and attempts to update the client fail. In this case, the zypper ref command shows an error like this:

```
Valid metadata not found at specified URL
```

To correct this, determine if Taskomatic is still in the process of generating repository metadata, or if a crash could have occurred. Wait for metadata regeneration to complete or restart Taskomatic after a crash in order for client updates to be carried out correctly.

Procedure: Resolving Taskomatic Problems

1. On the Uyuni Server, check the /var/log/rhn/rhn_taskomatic_daemon.log file to determine if any metadata regeneration processes are still running, or if a crash occurred.
2. Restart taskomatic:

```
service taskomatic restart
```

3. In the Taskomatic log files, you can identify the section related to metadata regeneration by looking for opening and closing lines that look like this:

```
<YYYY-DD-MM> <HH:MM:SS>,174 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Generating new repository
metadata for channel 'cloned-2018-q1-sles12-sp3-updates-x86_64'(sha256) 550 packages,
140 errata

...

<YYYY-DD-MM> <HH:MM:SS>,704 [Thread-584] INFO
com.redhat.rhn.taskomatic.task.repomd.RepositoryWriter - Repository metadata
generation for 'cloned-2018-q1-sles12-sp3-updates-x86_64' finished in 4 seconds
```

30.37. Troubleshooting Failed Token Refresh

The channel access tokens that Salt clients use to download packages from Uyuni expire and are replaced regularly. A Taskomatic job named TokenCleanup refreshes the tokens in the server database. By default, it then deploys the new credentials to the affected clients. This is controlled by the server.susemanager.token_refresh_auto_deploy setting. When the setting is disabled, the job only writes a

warning to the log that lists the affected clients, and no client receives the new tokens.

If a client is offline or unreachable when the job runs, the server database holds the new token but the client does not. The job does not retry on later runs, because it finds the tokens in the database valid.

When this happens, the client keeps outdated tokens in /etc/zypp/repos.d/susemanager:channels.repo. Running zypper ref or zypper up directly on the client, or using an external tool that does so, does not refresh the tokens and fails with an error like this:

```
Retrieving repository '<repository name>' metadata ....[error]
Repository '<repository name>' is invalid.
[susemanager:<channel label>|https://<server>:443/rhn/manager/download/<channel
label>?<token>] Valid metadata not found at specified URL
```

On the server, /var/log/rhn/rhn_web_ui.log records the rejected requests as INFO messages like Forbidden: Token … is not valid to access repomd.xml. These messages are only written while INFO logging is enabled for the com.suse.manager.webui.controllers.DownloadController logger, which is the default. The same Valid metadata not found message can also be caused by incomplete repository metadata regeneration. For more information, see **Administration Guide › Troubleshooting › Tshoot Taskomatic**.

To fix an affected client immediately, apply the channels state or a highstate to it. Any other channel-related action in Uyuni, such as changing the channel assignment or installing a package, also redeploys the current tokens.

To prevent the problem, schedule a recurring action for the organization that redeploys the certificates and channel configuration. Clients that were unavailable during a token refresh then receive the current tokens the next time the action runs.

Procedure: Keeping channel tokens synchronized

1. In the Uyuni Web UI, navigate to **Admin › Organizations**.
2. Click the name of the organization to configure.
3. Navigate to the Recurring Actions tab and click **[Create]**.
4. Select an action type from the Action Type dropdown:
    - Highstate applies the complete configuration of each client.
    - Custom state applies only the states you select.
5. Type a name for the schedule.
6. Choose a frequency, for example Weekly or Daily.

7. For actions of type Custom state, search for and select Sync All, Certificates, and Channels, which are the util.syncall, certs, and channels states, and click **[Save Changes]**.
8. In the next pane, drag and drop the states into that order and click **[Confirm]**.
9. Click **[Create Schedule]**.

For more information about recurring actions, see **Administration Guide › Recurring Actions**.



Repeat this procedure for every organization that contains clients that are not always online.



Only the Uyuni Administrator can open the organization from **Admin › Organizations**. Organization Administrators can create the same recurring action in **Home › My Organization › Recurring Actions**.

30.38. Troubleshooting Web UI Fails to Load

Sometimes, the Web UI will not load after migration. This is usually caused by browser caching, if the new system has the same hostname and IP address as the old system. This duplication can confuse some browsers.

This issue is resolved by clearing the cache and reloading the page. In most browsers, you can do this quickly by pressing `Ctrl` + `F5`.

Chapter 31. GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these

sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other

respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".