Инструкция по созданию дополнительных параметров групповых политик (шаблонов) в ALD Pro 2.2.0 и выше

ALD Pro

Exported on 03/07/2024

Table of Contents

Актуально для версии ALD Pro 2.2.0 и выше

1 Термины и определения

Термин	Определение
Дополнительные параметры групповых политик	Параметры групповой политики, сконфигурированные под определенные задачи, на основе скрипта SaltStack (Путь: Управление доменом → Дополнительные параметры групповых политик).
Salt minion	Сервер, на котором запущен демон Salt minion, который может прослушивать команды от мастера (Salt-master) и выполнять полученные задачи.
Standalone minion	Автономно от мастера запущенный клиент на компьютере домена. Выполняет задачи запуска заданий без подключений к мастеру и запуска локальных задач.
Salt-master	Центральный демон Salt, которой может отдавать команды слушающим миньонам (Salt-minion).
Приоритет политики	Позволяет управлять очередностью применения политик.
SaltStack	Система управления конфигурацией машин и инструмент дистанционного выполнения скриптов, который позволяет администраторам запускать команды на разных машинах.
Сервер службы каталогов	Сервер, хранящий и передающий важную конфиденциальную информацию, связанную с пользователями, паролями и учетными записями компьютеров.
LDAP-протокол	Протокол прикладного уровня для доступа к службе каталогов.
Портал управления	Графический web-интерфейс, предоставляющий привилегированному пользователю единую точку доступа к данным, управления инфраструктурой и объектами домена.
Политика	Правило, в соответствии с которым настраивается рабочая среда.

Групповые политики позволяют снизить стоимость управления ИТ-инфраструктурой за счет автоматического применения одинаковых настроек на больших группах компьютеров, имеющих общее целевое назначение.

Каждая групповая политика (в терминах MS объект групповой политики) представляет собой именованный набор параметров, в соответствии с которыми производится автоматическая настройка операционной системы и прикладного программного обеспечения. Для настройки групповой политики нужно выполнить следующие действия:

- создать новую групповую политику
- добавить конфигурационный параметр в политику (включить его)
- установить желаемые значения атрибутов для параметра
- назначить политику на подразделение, внутри которого есть целевые компьютеры или пользователи
- подождать от 30 минут до 80 минут или изменить таймер для получения и применения политики.

Администраторам доступны сотни параметров «из коробки», но для решения специфичных задач конкретного бизнеса может потребоваться разработка дополнительных параметров силами команды внедрения, и в ALD Pro для этого есть все необходимые инструменты.

2 Как работают групповые политики

Для получения данных групповых политик и их применения используется pull-модель. Инициатором работы является миньон SaltStack (standalone minion), установленный на каждом компьютере домена, который по протоколу LDAP к серверу службы каталогов получает актуальные данные о групповой политике.

- **Миньон Standalone** — это рабочие станции или серверы, на котором работает демон-миньон Salt, обладающий широкой функциональностью, работающий автономно от мастера. Используется для запуска заданий в системе без подключения к мастеру.

> **Примечание**
>
> Получение данных групповых политик и применение политик с помощью standalone миньона реализовано с версии ALD Pro 2.2.0, ранее для этих задач использовался стандартный миньон SaltStack, который получал данные от мастера.

3 Как создать дополнительный параметр

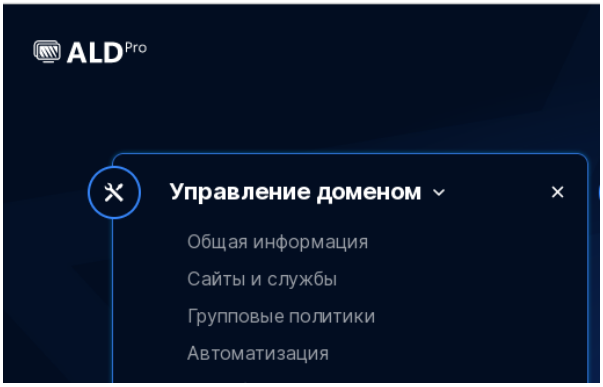
Параметр групповой политики (в терминах MS шаблон групповой политики) подобен классу из теории объектно-ориентированного программирования. Он определяет модель для создания объектов конфигурирования, описывая их свойства (атрибуты) и методы для работы с этими данными (скрипт). Продолжая аналогию из ООП, «объектом» в данном случае будет являться экземпляр параметра, добавленный в конкретную групповую политику.

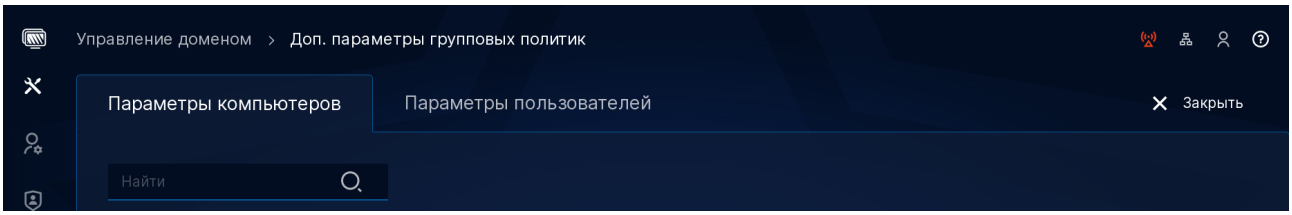
Параметр может быть «простым» или «составным»:

- В случае **простого** параметра свойства представляют из себя плоский список атрибутов. Например, для настройки межсетевого экрана может потребоваться задать уровень детализации журнала, и эта настройка подразумевает одно конкретное значение, поэтому она может быть реализована атрибутом «простого» параметра.
- **Составные** параметры позволяют задавать таблицу настроек, где в каждой строке представлен однотипный набор данных. Если продолжать пример с межсетевым экраном, то для настройки фильтрации может потребоваться разрешить входящие ssh-соединения, запретить исходящие smtp и т.п., поэтому такие настройки нужно реализовывать атрибутами «составного» параметра.

Параметр может быть «параметром компьютера» или «параметром пользователя». «Параметры компьютеров» отвечают за настройку оборудования и служб вне зависимости от того, какой пользователь работает с системой. «Параметры компьютеров» назначаются на компьютеры выбранного подразделения. «Параметры пользователей» отвечают за настройку рабочей среды пользователя вне зависимости от того, на каком компьютере он работает. «Параметры пользователей» назначаются на пользователей выбранного подразделения. В обоих случаях скрипты выполняются с возможностями по администрированию root-пользователя.

Чтобы создать новый параметр групповой политики вам нужно перейти на страницу «Управление доменом \ Доп. параметры групповых политик» и выбрать одну из вкладок:





На каждой вкладке по умолчанию будет представлен корневой каталог «Каталог дополнительных параметров», внутри которого вы можете создавать свои пользовательские параметры. Если параметров станет слишком много, вы сможете группировать их по папкам.

Выберите папку «Каталог дополнительных параметров», нажмите кнопку «+ Новый параметр» и заполните карточку следующими значениями:

- Раздел «Основные»
  - Название параметра: «Диспетчер задач htop»
    Это название, которое вы хотите видеть в каталоге при редактировании групповых политик на портале управления.
  - Уникальный идентификатор: «software_htop»
    *Это название, которое будет использоваться в качестве имени *.sls файла на диске и в скриптах SaltStack.*
  - Тип каталога: «Параметр компьютерной групповой политики»
    Определяет, относится ли данный параметр к настройкам компьютера или пользователя. Параметр не доступен для редактирования, его значение определяется автоматически в зависимости от того, с какой страницы вы перешли к созданию дополнительного параметра.
  - Тип параметра: «простой»
    Позволяет указать способ хранения атрибутов.
  - Папка параметра: «Дополнительные параметры»
    Указывает раздел каталога, в котором будет доступен данный параметр.
  - Назначение параметра: «Управление установкой и настройкой диспетчера задач htop»
    Текстовое описание, позволяющее указать наиболее важную информацию о работе параметра для удобства последующего использования. Данный текст отображается при редактировании групповой политики по нажатию кнопки вопроса
- Разделы «Атрибуты параметра» и «Конфигурация скрипта» станут доступны при редактировании сохраненного параметра.

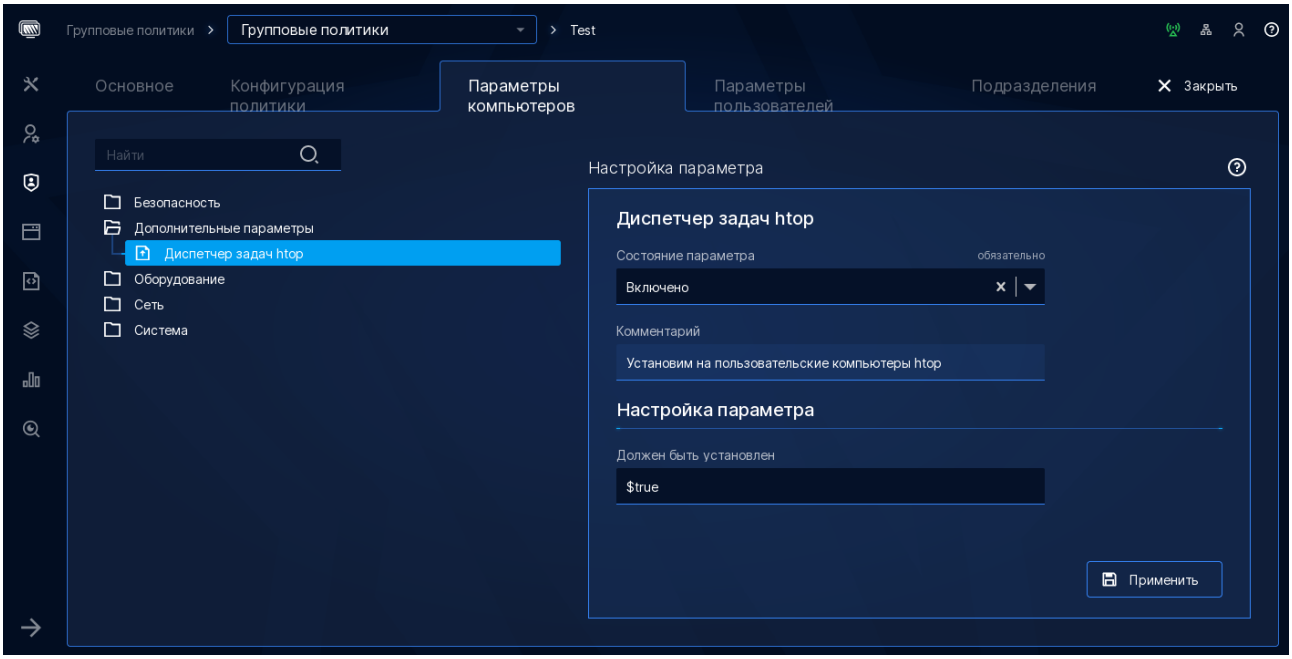
3.1 Особенности редактирования дополнительных параметров ГП

Для редактирования дополнительного параметра групповых политик нужно перейти на страницу «Управление доменом \ Доп. параметры групповых политик \ Параметры компьютеров или Параметры пользователей"Имя параметра"» и внести нужные изменения. После сохранить.

Чтобы эти параметры синхронизировались с групповыми политиками и применились на компьютерах и пользователях нужно перейти на карточку этих политик «Групповые политики \ Групповые политики \ "Имя политики"», внести изменения и сохранить. Изменения в групповой политике приведут к повышению ее версии. Только после этого изменения дополнительных параметров ГП будут применены на компьютерах и для пользователей. Или принудительно применить политику (подробнее читайте в разделе 7. Отладка)
Аналогичные действия необходимо применить если дополнительный параметр групповой политики был удален.

4 Как настроить атрибуты дополнительного параметра

Атрибуты — это свойства, которые можно задавать при добавлении параметра к групповой политике. Например, если мы создадим параметр для управления диспетчером задач htop на рабочих станциях, то нам понадобится атрибут «Должен быть установлен», чтобы в зависимости от его значения устанавливать или удалять указанное программное обеспечение.



Атрибуты представляют из себя обычные текстовые строки, значение которых может быть интерпретировано в скриптах в том числе как числа, логические значения ($true и $false), списки и т. п. Добавьте атрибут:

- Название атрибута: «Должен быть установлен»
  Название, под которым вы хотите видеть этот атрибут на карточке параметра.
- Уникальный идентификатор: «must_be_installed»
  Идентификатор атрибута, который будет использоваться как имя переменной в SaltStack скриптах.
- Описание: «атрибут используется как логическая переменная $true/$false для ветвления шаблона»
  Краткие комментарии для инженера, которые помогут упростить поддержку этого атрибута в будущем, когда потребуется внести какие-либо изменения. Данная информация недоступна при настройке групповой политики, поэтому, если вы хотите дать подсказку администратору о возможных значениях этого атрибута, внесите эту информацию в описание параметра.

5 Скрипт дополнительного параметра

Скрипты SaltStack похожи на PHP, в них текст перемежается императивными инструкциями языка программирования, по результату выполнения которых получается итоговый документ. За императивную часть отвечают инструкции шаблонизатора Jinja2, а декларативная часть задается по правилам SaltStack в YAML-подобном синтаксисе.

5.1 Императивные инструкции шаблонизатора Jinja

Инструкции языка Jinja внедряются с помощью фигурных скобок:

Jinja	**Аналог PHP**	**Комментарий**
{% … %}	<? … ?>	Синтаксис для вставки инструкции языка шаблонизатора Jinja
{% print(…) %}	<? echo (…) ?>	Пример инструкции для вывода в документ по месту вызова значения выражения
{{ … }}	<?= … ?>	Краткий синтаксис для вывода в документ по месту вызова значения выражения
{# … #}	<?/* … */?>	Синтаксис для вставки комментариев средствами языка Jinja2

Для повышения читабельности кода управляющие структуры обычно оформляют отступами, но yaml-документы крайне чувствительны к пробелам, поэтому данный инструмент форматирования оказывается недоступен без применения специальных операторов подавления отступов. Чтобы убрать лишние пробелы и пустые строки вам нужно всего лишь поставить знак дефиса слева, справа или с обоих концов управляющего блока:

{%-	Удаляет пробелы и пустые строки слева
-%}	Удаляет перенос текущей строки и переносит ее на предыдущую
{%- и -%}	Удаляет пробелы и пустые строки слева, в т.ч. перенос текущей строки, поэтому текст окажется в конце предыдущей строки

5.1.1 Выполнение команд

Создайте файл test.sls с простейшим Jinja-скриптом, чтобы проверить, как это работает, и выполните его с помощью команды salt-call с указанными параметрами:

```
# echo '{% do salt.log.info("Hello world!") %}' > hello.sls
# salt-call --local --file-root=. state.sls hello
```

Несколько комментариев:

- Оператор «do» вызывает метод salt.log.info с параметром 'Hello world!'
- Команда salt-call отвечает за локальное исполнение скриптов на миньонах.
- Параметр local исключает обращение к мастеру для получения настроек.
- Параметр file-root со значением «.» устанавливает в качестве рабочего каталога текущую директорию.
- Параметр state.sls указывает на то, что следует использовать метод sls модуля salt.modules.state, который применяет состояния, описанные в одном и более файлов на диске из рабочей директории.
- Параметр hello задает имя файла hello.sls, которое должно быть указано без расширения файла.

Приведем еще один пример. С помощью метода run модуля cmd вы можете выполнить на удаленном хосте любую команду bash:

```
# echo '{% do salt.cmd.run("apt-get install htop") %}' > installhtop.sls
# salt-call --local --file-root=. state.sls installhtop
```

Полный перечень всех модулей SaltStack можно найти на странице документации https://docs.saltproject.io/en/latest/ref/modules/all/index.html

5.1.2 Работа с переменными

Переменные на языке Jinja задаются оператором set. Вам доступен широкий перечень типов данных: булевые, числовые, текстовые, списки, кортежи, словари:

```
{% set boolean_val = True %}
{% set int_val = 777 %}
{% set string_val = 'hello' %}
{% set list_val = ['h', 'e', 'l', 'l', 'o'] %}
{% set tuple_val = ('h', 'e', 'l', 'l', 'o') %}
{% set dict_val = { b: True, i: 777, s: 'hello') %}
```

При работе с числовыми переменными доступны обычные математические операторы:

```
{% set a = 5 %}
{% set b = 10 %}
{% set a = a + b %}
{% set b = a - b %}
{% set a = a - b %}
```

Конкатенация срок возможна как специальным оператором «~», который предварительно конвертирует все значения в строки, так и обычным оператором сложения «+»:

```
{% set a = 5 %}
```

```
{% set b = 10 %}
{% set c = a ~ b %}
{% set d = 'World' %}
{% set e = 'Hello, ' + d %}
```

Для выполнения более сложных преобразований над переменными вам доступно множество функций, которые можно применять как методы через точку или в стиле конвейеров bash через символ вертикальной черты, за что их так же называют фильтрами:

```
{% set a = 'Hello'.upper() %}
{% set b = 'world' | upper %}
```

Полный перечень встроенных фильтров Jinja можно найти в документации https://jinja.palletsprojects.com/en/2.11.x/templates/#list-of-builtin-filters

5.1.3 Ветвления и циклы

Если выполнение набора команд должно зависеть от некоторого условия, то ветвление можно задать с помощью операторов if/elseif/else/endif:

```
{% if ram >= 2048 %}
...
{% elseif ram <= 4096 %}
...
{% else %}
...
{% endif %}
```

Циклы for являются аналогом конструкций типа foreach других языков программирования и предназначены для выполнения заданного набора инструкций применительно к каждому элементу из списка:

```
{% set users = ['ivanov', 'petrov', 'kuznetsov'] %}
{% for user in users %}
{% do salt.log.info(user.upper()) %}
{% endfor %}
```

Если вам нужно будет обработать данные словаря, воспользуйтесь методом items(), чтобы получить список его элементов:

```
{% set users = {'u1':'ivanov', 'u2':'petrov', 'u3':'kuznetsov'} %}
{% for id, user in users.items() %}
{% do salt.log.info(user.upper()) %}
{% endfor %}
```

Так как циклы Jinja работают только со списками, то для эмуляции обычных циклов со счетчиком вам нужно воспользоваться функцией range([min], max). Если передать этой функции один параметр, то

будет сгенерирован список с указанным количеством элементов, нумерация которых будет начинаться с нуля. Если передать два числа, то нумерация элементов будет в указанном диапазоне:

```
{% do salt.log.info('Обратный отсчет') %}
{% for i in range(0, 10) %}
{% do salt.log.info(10 - i) %}
{% endfor %}
```

5.1.4 Информация о целевой системе (grains)

При написании скриптов Jinja вы можете опираться на информацию о целевом хосте, на котором этот скрипт будет применен. Данная информация доступна в словаре grains, тут вы найдете информацию об операционной системе, памяти, дисках, настройках сетевых интерфейсов и многом другом. Например, используя ключ nodename можно получить имя хоста:

```
{% set node = salt['grains.get']('nodename') %}
```

Актуальное содержание словаря grains для конкретного хоста можно посмотреть с мастера утилитой salt, или с миньона утилитой salt-call:

```
# salt pc-1.ald.company.local grains.items
# salt-call grains.items
local:
     ----------
     astra_version:
     ----------
    distr:
        orel
    version:
        1.7.2
    ...
```

Более подробную информацию о модулях grains можно найти в документации по проекту: https://docs.saltproject.io/en/latest/ref/grains/all/index.html

5.1.5 Передача информации с мастера (pillar)

При настройке групповых политик администраторы задают значения атрибутов, которые доменным компьютерам передаются через механизм пилларов (salt pillar, соляной столб). Актуальное содержание словаря pillar можно посмотреть с мастера утилитой salt, или с миньона утилитой salt-call/.

Для компьютеров:

```
# salt-call pillar.get aldpro-hosts:pc-1.ald.company.local -c /srv/salt/standalone/
config
```

Для пользователей:

```
# salt-call pillar.get aldpro-users:<логин пользователя> -c /srv/salt/standalone/
config
```

Содержание словаря определяется тем, какие политики назначены конкретному компьютеру или пользователю (условный аналог GPResult в MS AD). Данные пиллара доступны только тем миньонам, для кого они предназначены, поэтому через атрибуты можно передавать в том числе конфиденциальную информацию, например пароли служебных учетных записей, сертификаты.

Если же вам нужно проверить, будет ли применена конкретная политика на хосте, вы можете применить состояние с параметром test=True. Если по результатам выполнения команды вы увидите сообщение "No changes needed to be made", значит система уже получила pillar с заданной ГП:

```
# salt-call state.apply rbta_ldap_env_vars_h test=True
...
          ID: /etc/bash.bashrc
    Function: file.blockreplace
      Result: True
     Comment: No changes needed to be made
...
```

Более подробную информацию о модулях pillar можно найти в документации по проекту: https://docs.saltproject.io/en/latest/topics/tutorials/pillar.html

5.1.6 Создание файлов на стороне миньона

Есть методы для создания файлов touch, переименования rename, удаления clean. Подробное описание возможностей доступно:

https://docs.saltproject.io/en/latest/ref/states/all/salt.states.file.html

https://www.8host.com/blog/osnovy-saltstack-bazovye-terminy-i-ponyatiya/

https://serverspace.ru/support/help/ispolzovanie-platformy-salt/

5.2 Декларативные описания SaltStack

Декларативная часть скрипта описывает в YAML-формате желаемую конфигурацию компьютера. В структуре документа указано, какие методы нужно вызывать для конфигурирования системы, и с какими параметрами. В качестве примера создадим файл software.sls со скриптом, который описывает состояние, после применения которого в системе должен стать доступен диспетчер задач htop:

```
# cat software.sls
software_htop: # Имя состояния
··pkg.installed: # Вызов метода installed модуля pkg
·····-·pkgs: # Аргументы метода installed
·······-·htop # Значение аргумента pkgs
```

Вы можете выполнить скрипт с помощью команды salt-call:

```
# salt-call --local –file-root=. state.sls software
```

Рассмотрим параметры команды подробнее:

- local — означает локальную обработку без обращения к мастеру
- file-root=. — устанавливает рабочую директорию, в которой будет выполняться поиск *.sls файлов
- state.sls — указывает, что следует использовать модуль sls, который отвечает за применение состояний из файлов на диске
- software — указывает имя *.sls файла без расширения

Язык YAML, также как и JSON, является языком разметки текста для сериализации данных. Каждая строка задает пару ключ-значение, между которыми стоит знак двоеточия. Ключи могут состоять из одного и более слов, причем, заключать их в кавычки необязательно. В качестве значений поддерживаются как скалярные типы данных (int, float, boolean, string), так и вложенные словари, что позволяет создавать древовидные структуры данных. Второй и последующий уровни дерева должны обозначаться отступами с помощью двух и более пробелов, использовать символы табуляции вместо пробелов недопустимо. Если требуется прокомментировать какой-то фрагмент документа, то слева от комментария нужно поставить символ решетки.

На верхнем уровне структуры данных должен быть указан ключ с именем состояния, в нашем случае это software_htop. Имя выбирается произвольно, в одном документе может быть описано несколько состояний, но их имена не должны повторяться.

На второй строке указано, что требуется вызвать метод installed модуля salt.states.pkg. Данный модуль является модулем состояния, т. е. его методы приводят систему к требуемому состоянию, описывая желаемый конечный результат, а не то, как к нему прийти. Есть также модули исполнения, которые выполняют в системе конкретные действия, ранее мы уже показывали пример с использованием метода run модуля cmd для выполнения bash команды.

На третьей и четвертой строке методу передается аргумент pkgs со значением htop. Обратите внимание, что аргумент и его значение являются элементами списков, поэтому в начале этих строк стоит символ дефиса.

6 Требования к скриптам и наследование параметров

С 2.2.0 архитектурой ALD Pro в части работы групповых политик изменилась, скрипты групповых политик получают только те компьютеры и пользователи, которым назначена политика. В связи с этим в скрипте может не быть проверки на назначение параметра конкретному компьютеру или пользователю.

Если на один компьютер или пользователя назначено несколько политик, использующих один и тот же параметр, то значения простых параметров будут переопределяться, а значения составных параметров суммироваться. Миньон получает политики, назначенные на компьютер или авторизованного на компьютере пользователя. ALD Pro анализирует назначенные политики, снизу вверх по дереву подразделений и формирует единый словарь pillar. Если на одно подразделение назначено несколько политик, их атрибуты обрабатываются в порядке указанных приоритетов. Если на компьютер или пользователя назначен один и тот же параметр групповой политики несколько раз через разные подразделения, наибольший приоритет будет у политики подразделения непосредственно содержащим пользователя или компьютер. Скрипты групповых политик отрабатываются миньоном только один раз с итоговым содержанием словаря pillar, вне зависимости от того, сколько раз соответствующий параметр был использован в групповых политиках, назначенных на этот компьютер или пользователя.

7 Отладка

Скрипт дополнительного параметра сохраняется в сервере службы каталогов ({корень} , cn=etc, cn=aldpro,cn=vcsStorage,cn=grouppolicy, cn={Идентификатор ГП}) и при получении по протоколу LDAP кешируется на миньоне в папках вместе с остальными групповыми политиками:
Параметры компьютеров: /srv/salt/standalone/roots/states/policies/host-policies/
Параметры пользователей: /srv/salt/standalone/roots/states/policies/user-policies/.

В соответствии с расписанием из файла /srv/salt/standalone/config/minion.d/standalone_scheduler.conf скрипты выполняются при запуске службы salt-minion и по расписанию каждые 30 минут + от 5 до 50 минут случайного времени. Другими словами политики гарантировано применятся в течение 80 минут. Посмотреть текущее значение таймера можно командой:

```
# salt-call schedule.show_next_fire_time build_and_run_gp -c /srv/salt/standalone/
config
```

При необходимости можно локально изменить время стационарного таймера (по умолчанию 30 минут):

1) В файле на клиентском компьютере: /srv/salt/standalone/config/minion.d/standalone_scheduler.conf изменить значение атрибута "seconds"/"minutes" в блоке интересующего задания. Для групповых политик это build_and_run_gp.

2) В файле /srv/salt/standalone/roots/_modules/utils.py заменяем _25_MINUTES = dt.timedelta(minutes=25). Выставлять значения таймера менее 10 минут нежелательно, это может привести к повышению нагрузки и на контроллер домена и на доменный компьютер.

3) Перезапускаем миньон командой:

```
# systemctl restart salt-minion-standalone.service
```

Другой способ, применения групповых политик можно на миньоне вручную ввести команду (аналог gpupdate /force):

```
# salt-call state.apply gpupdate.gp -c /srv/salt/standalone/config/
```

Можно в конце команды добавить pillar='{"force": True}' - принудительное удаление связанного с политикой кэша, pillar='{"verbose": True}' нужен для получения логов выполнения заданий применения. Использовать рекомендуется осторожно, поскольку вызывает повышенную нагрузку на контроллер домена.

Применить отдельно политик пользователя можно командой:

```
# salt-call state.apply policies.user-policies -c /srv/salt/standalone/config
pillar="{'user': 'username'}"
```

Для работы с политиками пользователей нужно явно передавать имя пользователя, чьи настройки вы хотите применить pillar="{'user': 'username'}".

Для отладки конкретного скрипта дополнительного параметра групповой политики компьютера можно использовать команду на стороне миньона:

```
# salt-call state.apply имя_скрипта -c /srv/salt/standalone/config
```

Для отладки конкретного скрипта дополнительного параметра групповой политики пользователя необходимо дополнительно явно указать пользователя, для которого запускается скрипт

```
# salt-call state.apply имя_скрипта -c /srv/salt/standalone/config pillar="{'user':
'username'}"
```

7.1 Версионность

Каждая политика ГП имеет версию, которая автоматически изменяется, если наступило одно из следующих событий:

1. Внесены любые изменения в политики ГП. В этом случае дата изменений - это дата, когда пользователь внес изменения. А автор - это ФИО пользователя, внесшего изменения.
2. Произошло обновление версии ALD Pro. В этом случае дата изменений - это дата обновления системы. А автора изменений - **Системное обновление**.