Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(УНИВЕРСИТЕТ ИТМО)

**УТВЕРЖДЕН**

**СНАБ.465600.001-ЛУ**

ПИЛОТНЫЙ УЧАСТОК МАГИСТРАЛЬНОЙ КВАНТОВОЙ СЕТИ (ПУ МКС)

Пилотный участок магистральной квантовой сети (ПУ МКС)  
(шифр «7.422»)

Инструкция по формированию и ведению базы данных (набора данных)

**СНАБ.465600.001 И4**

**Листов 56**

Содержание

[1 Правила подготовки данных](#_Toc73371828) 4

[2 Порядок и средства заполнения базы данных](#_Toc73371832) 43

[3 Процедуры изменения и контроля базы данных](#_Toc73371836) 48

[4 Порядок и средства восстановления базы данных](#_Toc73371841) 50

[Перечень принятых сокращений](#_Toc73371866) 55

1. Правила подготовки данных
   1. БД подсистемы мониторинга

Отбор информации для включения в базу данных «zabbix» подсистемы мониторинга осуществляется заимствованной системой Zabbix путем автоматического опроса технических средств, входящих в состав ПУ МКС.

Ввод и корректировка данных мониторинга осуществляется автоматически только программными компонентами СУМ КС. Прямой доступ администраторов к базе данных не предполагается.

Правила подготовки и кодирования информации задаются системным администратором с помощью системы Zabbix путем настройки шаблонов Zabbix и присоединения их к хостам (узлам сети) через веб-интерфейс. В шаблонах Zabbix задаются параметры элементов данных, триггеров, графиков, правил низкоуровневого обнаружения, групп элементов данных, комплексных экранов и веб-сценариев.

Изменение длительности хранения данных мониторинга задается в веб-интерфейсе подсистемы мониторинга:

* в диалоге свойств элемента данных;
* при массовом обновлении элементов данных;
* при настройке задач очистки истории.

БД подсистемы мониторинга представляет собой типовую БД для системы Zabbix. Для хранения исторических данных мониторинга предназначены следующие таблицы БД Zabbix:

* history – числовые данные с плавающей запятой;
* history\_uint – целые числа;
* history\_str – строковые данные до 255 символов;
* history\_log – данные типа логов;
* history\_text – текстовые данные.

За хранение динамики изменений предназначены две таблицы:

* trends – числовые данные с плавающей запятой;
* trends\_uint – целые числа.

Динамика изменений является встроенным механизмом обработки данных истории, где для каждого часа хранятся значения минимума, максимума и усредненное значения, а также общее количество значений за этот час.

Для хранения электронного журнала событий предназначена таблица «events» БД Zabbix.

Информация представляется через веб-интерфейс АРМ на вкладке «Мониторинг».

Порядок внесения изменений информации описан в документах:

* Руководство по организации сопровождения;
* Руководство пользователя;
* Руководство системного программиста.

А также в документации Zabbix, расположенной на сайте по адресу https://www.zabbix.com/documentation/5.0/.

* 1. БД подсистемы технического учета

Отбор информации для включения в базу данных «glpi» подсистемы технического учета осуществляется заимствованной системой GLPI путем автоматического опроса технических средств, входящих в состав ПУ МКС.

Ввод и корректировка данных осуществляется только программными компонентами СУМ КС. Прямой доступ администраторов к базе данных не предполагается.

Правила подготовки и кодирования информации задаются системным администратором в полуавтоматическом режиме с помощью системы GLPI путем занесения описания типов и моделей оборудования, которым автоматически присваивается уникальный ID объекта.

БД подсистемы технического учета представляет собой типовую БД для системы GLPI. Для хранения данных технического учета предназначены следующие таблицы БД GLPI:

* glpi\_computers – для хранения данных о серверах и компьютерах;
* glpi\_networkequipments для хранения данных о коммутаторах и других сетевых устройствах;
* glpi\_racks – для хранения данных о стойках;
* glpi\_enclosures – для хранения данных о инженерном оборудовании;
* glpi\_pdus – для хранения данных о ИБП и управляемых розетках;
* glpi\_passivedcequipments – для хранения данных о CWDM;
* glpi\_networkports – для хранения данных о сетевых портах;
* glpi\_softwares – для хранения данных программном обеспечении;
* glpi\_logs – для хранения всех вносимых изменений (создание объекта, старое/новое значение при изменении объекта);
* glpi\_problems – для хранения данных о проблемах;
* glpi\_tickets – для хранения данных о заявках;
* glpi\_events – для хранения данных о событиях в системе (авторизация пользователя, изменения данных объекта пользователем и т.д.);
* glpi\_crontasklogs – для хранения данных журнала автоматических действий системы;
* glpi\_plugin\_fusioninventory\_\* – для хранения данных плагина Fusioninventory, который выполняет автоматическую инвентаризацию оборудования;
* и другие (смотреть документацию GLPI, расположенную на сайте по адресу https://glpi-developer-documentation.readthedocs.io/en/latest/devapi/database/).

Информация представляется через веб-интерфейс АРМ на вкладке «Технический учет».

Порядок внесения изменений информации описан в документах:

* Руководство по организации сопровождения;
* Руководство пользователя;
* Руководство системного программиста.

А также в документации GLPI, расположенной на сайте по адресу https://glpi-user-documentation.readthedocs.io/fr/latest/.

* 1. Сервисные БД СУМ КС

Отбор информации для включения в сервисные базы данных СУМ КС осуществляется сервисами СУМ КС.

Ввод и корректировка данных осуществляется через программные компоненты СУМ КС. Прямой доступ администраторов к базе данных не предполагается.

Правила подготовки и кодирования информации задаются системным администратором в полуавтоматическом режиме с помощью ПО АРМ путем создания сеансов передачи данных и передачи ключей, которым автоматически присваивается уникальный ID.

Информация представляется через веб-интерфейс АРМ.

Порядок внесения изменений информации описан в документах:

* Руководство по организации сопровождения;
* Руководство пользователя;
* Руководство системного программиста.

Описанные далее БД находятся на единой отдельной ВМ «postgres-N» (где N - номер ЦУМа) с установленной PostgreSQL. Базы данных DNS и DHCP не требуют изменений в процессе эксплуатации.

* + 1. Таблицы БД ПО АРМ.

Таблицы БД «arm» ПО АРМ представляют собой хранилище объектов системы DJANGO (<https://www.djangoproject.com>):

1. Таблица администраторов АРМ, в основном соответствует пользователям из LDAP.

TABLE accounts\_customuser

column\_name | data\_type

----------------------+-------------------------

id | integer

password | character varying

last\_login | timestamp with time zone

is\_superuser | boolean

first\_name | character varying

last\_name | character varying

email | character varying

is\_staff | boolean

is\_active | boolean

date\_joined | timestamp with time zone

username | character varying

container | boolean

key | character varying

settings | character varying

Поля:

* password – зашифрованный пароль
* last\_login – дата последнего захода
* is\_superuser – является ли пользователь с правами на все действия
* first\_name – имя
* last\_name – фамилия
* email – почта
* is\_staff – может ли зайти в административный интерфейс django
* is\_active – не заблокирован ли пользователь
* date\_joined – дата заведения пользователя в системе
* username – логин пользователя
* container – заведен ли дня него ключевой контейнер
* key – ключ
* settings – JSON строка настроек пользователя

1. Таблица для описания групп пользователей

TABLE accounts\_customuser\_groups

column\_name | data\_type

----------------------+----------

id | integer

customuser\_id | integer

group\_id | integer

Поля:

* customer\_id – идентификатор пользователя
* group\_id – идентификатор группы

1. Таблица для описания прав пользователей

TABLE accounts\_customuser\_user\_permissions

column\_name | data\_type

----------------------+----------

id | integer

customuser\_id | integer

permission\_id | integer

Поля:

* customer\_id – идентификатор пользователя
* permission\_id – идентификатор права

1. Таблица истории таблицы администраторов АРМ (accounts\_customuser).

TABLE accounts\_historicalcustomuser

column\_name | data\_type

------------------------------+-------------------------

id | integer

password | character varying

last\_login | timestamp with time zone

is\_superuser | boolean

first\_name | character varying

last\_name | character varying

email | character varying

is\_staff | boolean

is\_active | boolean

date\_joined | timestamp with time zone

username | character varying

container | boolean

key | character varying

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

history\_user\_id | integer

settings | character varying

Все таблицы с именем «historical» – это таблицы, соответствующие изменениям записей основной таблицы.

Дополнительные к основной таблице (в данном случае – accounts\_customuser) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица запросов на регистрацию пользователя/администратора АРМ.

TABLE accounts\_registrationapplication

column\_name | data\_type

----------------------+-------------------------

id | integer

username | character varying

email | character varying

contact | character varying

datetime | timestamp with time zone

processed | boolean

is\_active | boolean

groups | character varying

key | character varying

Поля:

* username – логин пользователя
* email – почтовый адрес пользователя
* contact – контактные данные
* datetime – дата запроса на регистрацию
* processed – статус, обработана ли заявка
* is\_active – статус, находится ли еще в обработке
* groups – группы пользователя
* key – ключ пользователя

1. Таблица учета всех действий пользователя в системе.

TABLE activity\_log\_activitylog

column\_name | data\_type

----------------------+-------------------------

id | integer

user\_id | integer

user | character varying

request\_url | character varying

request\_method | character varying

response\_code | character varying

datetime | timestamp with time zone

ip\_address | inet

extra\_data | text

Поля:

* user\_id – идентификатор пользователя
* user – логин пользователя
* request\_url – адрес страницы АРМ
* request\_method – тип обращения к странице
* datetime – дата обращения к странице
* ip\_address – с какого IP произошло обращение
* extra\_data – пустое поле, на будущее

1. Таблица групп администраторов.

TABLE auth\_group

column\_name | data\_type

----------------------+------------------

id | integer

name | character varying

Поля:

* name – имя группы

1. Таблица соответствия прав по группам.

TABLE auth\_group\_permissions

column\_name | data\_type

----------------------+----------

id | integer

group\_id | integer

permission\_id | integer

Поля:

* group\_id – идентификатор группы
* permission\_id – идентификатор права

1. Таблица прав пользователей.

TABLE auth\_permission

column\_name | data\_type

----------------------+------------------

id | integer

name | character varying

content\_type\_id | integer

codename | character varying

Поля:

* name – описание права
* content\_type\_id – идентификатор типа объектов
* codename – идентификатор права в системе

1. Таблица модуля AXES, отвечающего за ограничение количества неуспешных логинов. Перечисляет количество неудачных попыток захода для пользователя.

TABLE axes\_accessattempt

column\_name | data\_type

------------------------------+-------------------------

id | integer

user\_agent | character varying

ip\_address | inet

username | character varying

http\_accept | character varying

path\_info | character varying

attempt\_time | timestamp with time zone

get\_data | text

post\_data | text

failures\_since\_start | integer

Поля:

* user\_agent – поле из HTTP запроса
* ip\_address – IP адрес запроса
* username – имя пользователя
* http\_accept – поле из HTTP запроса
* attempt\_time – время запроса
* path\_info – URL запроса
* get\_data – строка запроса после URL
* post\_data – данные в запросе
* failures\_since\_start – количество неудачных попыток

1. Таблица модуля AXES, отвечающего за ограничение количества неуспешных логинов. Перечисляет все заходы в систему.

TABLE axes\_accesslog

column\_name | data\_type

----------------------+-------------------------

id | integer

user\_agent | character varying

ip\_address | inet

username | character varying

http\_accept | character varying

path\_info | character varying

attempt\_time | timestamp with time zone

logout\_time | timestamp with time zone

Поля:

* user\_agent – поле из HTTP запроса
* ip\_address – IP адрес запроса
* username – имя пользователя
* http\_accept – поле из HTTP запроса
* attempt\_time – время захода в систему
* path\_info – URL запроса
* logout\_time – время выхода из системы

1. Таблица глобальных настроек АРМ, пары ключ-значение.

TABLE core\_globalsettings

column\_name | data\_type

----------------------+-------------------------

id | uuid

key | character varying

value | character varying

datetime | timestamp with time zone

Поля:

* key – наименование настройки
* value – значение настройки
* datetime – время последнего изменения

1. Таблица истории глобальных настроек АРМ.

TABLE core\_historicalglobalsettings

column\_name | data\_type

------------------------------+-------------------------

id | uuid

key | character varying

value | character varying

datetime | timestamp with time zone

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

history\_user\_id | integer

Дополнительные к основной таблице (core\_globalsettings) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица коммутаторов.

TABLE data\_switch

column\_name | data\_type

----------------------+------------------

id | uuid

logical\_type | character varying

ports | smallint

control\_port | character varying

test\_port | character varying

description | character varying

operate | boolean

node\_id | integer

name | character varying

glpi\_id | character varying

community | character varying

ip | character varying

location | character varying

up\_id | uuid

Поля:

* id – идентификатор коммутатора
* logical\_type – тип коммутатора
* ports – число портов коммутатора
* control\_port – порт управления
* test\_port – порт тестирования
* description – описание
* operate – статус, рабочий
* node\_id – идентификатор узла (ОУ)
* name – имя коммутатора (из GLPI)
* glpi\_id – идентификатор коммутатора в GLPI
* community – настройки рутинга
* ip – IP коммутатора (из GLPI)
* location – расположение (из GLPI)
* up\_id – идентификатор верхнестоящего коммутатора

1. Таблица истории коммутаторов.

TABLE data\_historicalswitch

column\_name | data\_type

------------------------------+-------------------------

id | uuid

logical\_type | character varying

ports | smallint

control\_port | character varying

test\_port | character varying

description | character varying

operate | boolean

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

history\_user\_id | integer

node\_id | integer

name | character varying

glpi\_id | character varying

community | character varying

ip | character varying

location | character varying

up\_id | uuid

Дополнительные к основной таблице (data\_switch) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица клиентских соединений на коммутаторе.

TABLE data\_switchcustomerconnection

column\_name | data\_type

----------------------+------------------

id | uuid

port | character varying

customer | character varying

switch\_id | uuid

booked | boolean

customer\_id | uuid

description | character varying

dscp | character varying

ip | character varying

opt82 | character varying

vlan | smallint

Поля:

* id – идентификатор соединения
* port – имя порта
* customer – имя клиента
* switch\_id – идентификатор коммутатора
* booked – статус, используется ли сеансе связи
* customer\_id – идентификатор клиента
* description – описание
* dscp – настройки рутинга
* ip – IP, если в сеансе связи
* opt82 – настройки рутинга
* vlan – номер VLAN

1. Таблица – лог действий в административном интерфейсе DJANGO.

TABLE django\_admin\_log

column\_name | data\_type

----------------------+-------------------------

id | integer

action\_time | timestamp with time zone

object\_id | text

object\_repr | character varying

action\_flag | smallint

change\_message | text

content\_type\_id | integer

user\_id | integer

Поля:

* action\_time – время действия
* object\_id – идентификатор объекта
* object\_reprt – строковое представление объекта
* action\_flag – тип действия (1,2,3 – создание, изменение, удаление)
* change\_message – JSON c данными изменения
* content\_type\_id – тип объекта (точнее, идентификатор content\_type)
* user\_id – идентификатор пользователя

1. Таблица моделей DJANGO.

TABLE django\_content\_type

column\_name | data\_type

----------------------+------------------

id | integer

app\_label | character varying

model | character varying

Поля:

* app\_label – имя приложения джанго
* model – наименование класса объекта в джанго

1. Таблица миграций DJANGO.

TABLE django\_migrations

column\_name | data\_type

----------------------+-------------------------

id | integer

app | character varying

name | character varying

applied | timestamp with time zone

Поля:

* app – имя приложения джанго
* name – наименование миграции джанго
* applied – время, когда миграция была произведена

1. Таблица – список сессий DJANGO.

TABLE django\_session

column\_name | data\_type

----------------------+-------------------------

session\_key | character varying

session\_data | text

expire\_date | timestamp with time zone

Поля:

* session\_key – идентификатор сессии
* session\_data – данные сессии
* expire\_date – время окончания сессии

1. Таблица ОУ.

TABLE gws\_gw

column\_name | data\_type

----------------------+------------------

id | integer

name | character varying

description | character varying

address | character varying

state | character varying

log | character varying

Поля:

* name – имя ОУ
* description – описание ОУ
* address – URL доступа к ОУ
* state – состояние ОУ
* log – дополнительные данные

1. Таблица истории ОУ.

TABLE gws\_historicalgw

column\_name | data\_type

------------------------------+-------------------------

id | integer

name | character varying

description | character varying

address | character varying

state | character varying

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

history\_user\_id | integer

Дополнительные к основной таблице (gws\_gw) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица модуля CAS для централизированной аутентификации пользователей.

TABLE mama\_cas\_serviceticket

column\_name | data\_type

----------------------+-------------------------

id | integer

ticket | character varying

expires | timestamp with time zone

consumed | timestamp with time zone

service | character varying

primary | boolean

user\_id | integer

Поля:

* ticket – идентификатор сессии
* expires – конец сессии (время)
* consumed – начало сессии (время)
* service – сервер, который запросил аутентификацию
* primary – основной
* user\_id – идентификатор пользователя

1. Таблица «ФПСУ-IP» 10G.

TABLE products\_product

column\_name | data\_type

----------------------+------------------

id | integer

name | character varying

description | character varying

address | character varying

state | character varying

geo | character varying

operate | boolean

log | character varying

Поля:

* name – имя «ФПСУ-IP» 10G
* description – описание «ФПСУ-IP» 10G
* address – URL доступа к «ФПСУ-IP» 10G
* state – состояние «ФПСУ-IP» 10G
* geo – расположение «ФПСУ-IP» 10G (координаты)
* operate – состояние, в работе/не в работе
* log – дополнительные данные

1. Таблица истории «ФПСУ-IP» 10G.

TABLE products\_historicalproduct

column\_name | data\_type

------------------------------+-------------------------

id | integer

name | character varying

description | character varying

address | character varying

state | character varying

geo | character varying

operate | boolean

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

history\_user\_id | integer

Дополнительные к основной таблице (products\_product) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица сеансов передачи данных.

TABLE services\_dataservice

column\_name | data\_type

------------------------------+------------------

id | uuid

order | character varying

port1 | character varying

port2 | character varying

client | character varying

vlan | smallint

bandwidth | integer

gw1\_id | uuid

gw2\_id | uuid

status | character varying

connection\_information | character varying

Поля:

* order – идентификатор заказа
* port1 – порт А сеанса
* port2 – порт Б сеанса
* client – имя клиента
* vlan – номер VLAN
* bandwidth – пропускная способность сеанса
* gw1\_id – идентификатор коммутатора А
* gw2\_id – идентификатор коммутатора Б
* status – состояние сеанса
* connection\_information – информация по соединению (от модуля создания каналов)

1. Таблица истории сеансов передачи данных.

TABLE services\_historicaldataservice

column\_name | data\_type

------------------------------+-------------------------

id | uuid

order | character varying

port1 | character varying

port2 | character varying

client | character varying

vlan | smallint

bandwidth | integer

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

gw1\_id | uuid

gw2\_id | uuid

history\_user\_id | integer

status | character varying

connection\_information | character varying

Дополнительные к основной таблице (services\_dataservice) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)

1. Таблица сеансов передачи ключей.

TABLE services\_keyservice

column\_name | data\_type

------------------------------+------------------

id | uuid

order | character varying

port1 | character varying

port2 | character varying

client | character varying

vlan | smallint

key\_length | integer

key\_change | integer

gw1\_id | uuid

gw2\_id | uuid

status | character varying

connection\_information | character varying

Поля:

* order – идентификатор заказа
* port1 – порт А сеанса
* port2 – порт Б сеанса
* client – имя клиента
* vlan – номер VLAN
* key\_length – длина ключа
* key\_change – скорость изменения ключа
* gw1\_id – идентификатор коммутатора А
* gw2\_id – идентификатор коммутатора Б
* status – состояние сеанса
* connection\_information – информация по соединению (от модуля создания каналов)

1. Таблица истории сеансов передачи ключей.

TABLE services\_historicalkeyservice

column\_name | data\_type

------------------------------+-------------------------

id | uuid

order | character varying

port1 | character varying

port2 | character varying

client | character varying

vlan | smallint

key\_length | integer

key\_change | integer

history\_id | integer

history\_date | timestamp with time zone

history\_change\_reason | character varying

history\_type | character varying

gw1\_id | uuid

gw2\_id | uuid

history\_user\_id | integer

status | character varying

connection\_information | character varying

Дополнительные к основной таблице (services\_keyservice) поля:

* history\_date – дата изменения
* history\_change\_reason – причина
* history\_type – +,-,~ – создание, удаление, изменение
* history\_user\_id – идентификатор пользователя, внесшего изменения (если пуст, изменение произошло автоматически на celery)
  + 1. Таблицы БД подсистемы взаимодействия с внешними системами.

Таблицы БД «pum\_outss» программы взаимодействия СУМ КС с ОУТСС служат для установления соответствия объектов системы мониторирования Zabbix в СУМ КС с объектами в ОУТСС.

1. Таблица events\_table

Таблица events\_table ставит в соответствие событие (проблема) в СУМ КС с событием в ОУТСС.

column\_name | data\_type

----------------------+-----------------------------

pum\_event\_id | integer

host\_id | integer

service\_id | integer

state | integer

dt\_update | timestamp without time zone

check\_dt | timestamp without time zone

eventcmdb\_id | integer

outssbody | json

plugin\_output | text

Назначение полей:

* pum\_event\_id – идентификатор события в СУМ КС (Zabbix)
* host\_id – идентификатор хоста, связанного с событием
* service\_id – идентификатор сервиса (триггера), связанного с событием
* state – состояние события, 1 – проблема, 0 – проблема разрешена
* dt\_update – дата последнего обновления
* check\_dt – дата возникновения проблемы
* eventcmdb\_id – идентификатор события в ОУТСС
* outssbody – содержание учетной карточки (УК) события в ОУТСС
* plugin\_output – вывод проверки в Zabbix

1. Таблица hosts\_table

Таблица hosts\_table ставит в соответствие оборудование в СУМ КС (Zabbix) с оборудованием в ОУТСС.

column\_name | data\_type

----------------------+-----------------------------

host\_id | integer

equipmentcard\_id | integer

outssbody | json

pumbody | json

dt\_update | timestamp without time zone

Назначение полей:

* host\_id - идентификатор хоста в Zabbix
* equipmentcard\_id – идентификатор экземпляра оборудования в ОУТСС
* outssbody – содержание учетной карточки (УК) оборудования в ОУТСС
* pumbody – информация о хосте из Zabbix, связанного с оборудованием в ОУТСС
* dt\_update – дата последнего изменения

1. Таблица services\_table

Таблица services\_table содержит информацию о сервисах (триггерах).

column\_name | data\_type

----------------------+-----------------------------

service\_id | integer

pumbody | json

dt\_create | timestamp without time zone

dt\_update | timestamp without time zone

Назначение полей:

* service\_id – идентификатор сервиса(триггера) в Zabbix
* pumbody – информация о сервисе в Zabbix, отправляемая в ОУТСС
* dt\_create – дата создания сервиса
* dt\_update – дата последнего изменения
  + 1. Таблицы БД модуля управления ТТТ.

В МУ ТТТ используются 2 базы данных -- dhcplog и radethdb.

* + - 1. БД dhcplog

БД dhcplog создается с помощью следующих SQL–инструкций:

-- CREATE DATABASE dhcplog WITH OWNER 'stdhcp';

CREATE TYPE msgs AS ENUM ('DHCP-Discover', 'DHCP-Request', 'another');

1. Таблица log

Таблица log используется системой FreeRADIUS, данные заносятся автоматически при выполнении каждого запроса. Таблица log создается с помощью следующих SQL–инструкций:

CREATE TABLE log (

id serial,

dt timestamp with time zone DEFAULT now() NOT NULL,

mac macaddr NOT NULL,

switch char(12) DEFAULT NULL,

port integer DEFAULT NULL,

vlan integer DEFAULT NULL,

ip inet NOT NULL,

msgtype msgs NOT NULL DEFAULT 'another',

extra varchar(45) DEFAULT NULL

);

CREATE INDEX log\_switch ON log (

switch bpchar\_pattern\_ops,

port int4\_ops

);

CREATE INDEX log\_mac ON log (

mac macaddr\_ops,

dt timestamp\_ops

);

CREATE INDEX log\_dt ON log (

dt timestamp\_ops,

mac macaddr\_ops

);

CREATE INDEX log\_msgtype ON log (msgtype);

CREATE INDEX log\_ip ON log (

ip inet\_ops,

switch bpchar\_pattern\_ops,

port int4\_ops

);

1. Таблица logindex

Таблица logindex используется системой FreeRADIUS, данные заносятся автоматически при появлении каждой записи в таблице log. Таблица logindex создается с помощью следующих SQL–инструкций:

CREATE TABLE logindex (

id serial,

tablename varchar(32) NOT NULL,

startts timestamp with time zone NOT NULL,

endts timestamp with time zone NOT NULL

);

* + - 1. БД radethdb

БД radethdb создается с помощью следующих SQL–инструкций:

-- CREATE DATABASE radethdb WITH OWNER 'stdhcp';

1. Таблица IpPortBinding

Таблица IpPortBinding используется системой FreeRADIUS, данные заносятся и удаляются автоматически при выполнении запросов. Таблица IpPortBinding создается с помощью следующих SQL–инструкций:

DROP TABLE IF EXISTS IpPortBinding;

CREATE TABLE IpPortBinding (

ip inet NOT NULL,

mask inet DEFAULT NULL,

gw inet DEFAULT NULL,

SwitchMAC macaddr NOT NULL DEFAULT 'aa:0:0:0:1:0',

SwitchPort integer DEFAULT NULL,

SwitchVlan integer DEFAULT NULL

);

DROP INDEX IF EXISTS ipb\_ip;

CREATE INDEX ipb\_ip ON IpPortBinding (ip);

DROP INDEX IF EXISTS ipb\_ip\_switch;

CREATE UNIQUE INDEX ipb\_ip\_switch ON IpPortBinding using btree (

ip inet\_ops,

SwitchMAC macaddr\_ops

);

DROP INDEX IF EXISTS ipb\_port;

CREATE UNIQUE INDEX ipb\_port ON IpPortBinding using btree (

SwitchPort int4\_ops,

SwitchMAC macaddr\_ops,

SwitchVlan int4\_ops

);

1. Таблица ports

Таблица ports используется процедурой amqp-rd.pl, данные заносятся путём реплицирования из базы АРМ. Таблица ports создается с помощью следующих SQL–инструкций:

DROP TABLE IF EXISTS ports;

CREATE TABLE ports (

port\_id uuid NOT NULL,

port\_name text NOT NULL,

sw\_id uuid NOT NULL,

up\_id uuid NOT NULL,

sw\_ip inet,

up\_ip inet,

up\_name text NOT NULL,

opt82 text,

vlan integer NOT NULL,

dscp integer,

description text,

cl\_id uuid

);

1. Таблица switches

Таблица switches используется процедурой amqp-rd.pl, данные заносятся путём реплицирования из базы АРМ. Таблица switches создается с помощью следующих SQL–инструкций:

DROP TABLE IF EXISTS switches;

CREATE TABLE switches (

sw\_id uuid NOT NULL,

sw\_name text NOT NULL,

sw\_ip inet,

up\_id uuid,

up\_ip inet,

community text,

location text,

description text

);

1. Таблица sessions

Таблица sessions используется процедурой amqp-rd.pl, данные заносятся при создании запроса на создание сценария передачи данных. Таблица sessions создается с помощью следующих SQL–инструкций:

DROP TABLE IF EXISTS sessions;

CREATE TABLE sessions (

ses\_id uuid NOT NULL,

client\_id uuid NOT NULL,

created\_at timestamp DEFAULT NULL,

updated\_at timestamp DEFAULT NULL,

deleted\_at timestamp DEFAULT NULL,

expired\_at timestamp DEFAULT NULL,

ses\_ip inet,

port\_id uuid NOT NULL,

opt82 text,

vlan integer NOT NULL,

acl\_id integer NOT NULL,

ext\_acl\_id integer DEFAULT NULL,

comment text

);

1. Таблица ip\_pool

Таблица ip\_pool используется процедурой amqp-rd.pl, данные заносятся при создании IP инфраструктуры данного ЦУМ. Таблица ip\_pool создается с помощью следующих SQL–инструкций:

DROP TABLE IF EXISTS ip\_pool;

CREATE TABLE ip\_pool (

ip\_snet inet NOT NULL,

ip\_gw inet NOT NULL,

ip\_cl inet NOT NULL,

mask inet NOT NULL,

rmask inet NOT NULL,

vlan integer DEFAULT NULL

);

1. Таблица ext\_acls

Таблица ext\_acls используется процедурой amqp-rd.pl, данные заносятся при создании запроса на создание сценария передачи данных. Таблица ext\_acls создается с помощью следующих SQL–инструкций:

DROP SEQUENCE IF EXISTS ext\_acl\_id\_seq;

CREATE SEQUENCE ext\_acl\_id\_seq

INCREMENT BY 1

MAXVALUE 999999

NO MINVALUE

CACHE 1;

SELECT pg\_catalog.setval('ext\_acl\_id\_seq', 1, true);

DROP TABLE IF EXISTS ext\_acls;

CREATE TABLE ext\_acls (

ext\_acl\_id integer DEFAULT nextval('ext\_acl\_id\_seq') NOT NULL,

ext\_acl\_name text NOT NULL,

ip inet[] NOT NULL,

rmask inet[] NOT NULL,

cls\_id integer[] NOT NULL

);

1. Таблица acls

Таблица acls используется процедурой amqp-rd.pl, данные заносятся при создании IP инфраструктуры данного ЦУМ. Таблица acls создается с помощью следующих SQL–инструкций:

DROP SEQUENCE IF EXISTS acl\_id\_seq;

CREATE SEQUENCE acl\_id\_seq

INCREMENT BY 1

MAXVALUE 999999

NO MINVALUE

CACHE 1;

SELECT pg\_catalog.setval('acl\_id\_seq', 1, true);

DROP TABLE IF EXISTS acls;

CREATE TABLE acls (

acl\_id integer DEFAULT nextval('acl\_id\_seq') NOT NULL,

acl\_name text NOT NULL,

ip inet NOT NULL,

rmask inet NOT NULL,

cls\_id integer NOT NULL

);

1. Таблица classes

Таблица classes используется процедурой amqp-rd.pl, данные заносятся при создании IP инфраструктуры данного ЦУМ. Таблица classes создается с помощью следующих SQL–инструкций:

DROP SEQUENCE IF EXISTS cl\_id\_seq;

DROP SEQUENCE IF EXISTS cls\_id\_seq;

CREATE SEQUENCE cls\_id\_seq

INCREMENT BY 1

MAXVALUE 999999

NO MINVALUE

CACHE 1;

SELECT pg\_catalog.setval('cls\_id\_seq', 1000, true);

DROP TABLE IF EXISTS classes;

CREATE TABLE classes (

cls\_id integer DEFAULT nextval('cls\_id\_seq') NOT NULL,

cls\_name text NOT NULL,

bandwidth integer NOT NULL,

priority integer,

pol\_id integer NOT NULL,

comment text

);

1. Таблица policies

Таблица policies используется процедурой amqp-rd.pl, данные заносятся при создании IP инфраструктуры данного ЦУМ. Таблица policies создается с помощью следующих SQL–инструкций:

DROP SEQUENCE IF EXISTS pol\_id\_seq;

CREATE SEQUENCE pol\_id\_seq

INCREMENT BY 1

MAXVALUE 999999

NO MINVALUE

CACHE 1;

SELECT pg\_catalog.setval('pol\_id\_seq', 1000, true);

DROP TABLE IF EXISTS policies;

CREATE TABLE policies (

pol\_id integer DEFAULT nextval('pol\_id\_seq') NOT NULL,

pol\_name text NOT NULL,

comment text

);

* + - 1. Описание структуры канала связи и подходов деления по полосам.
  + Ширина общей полосы связи транспортной сети устанавливается шириной в 10Gb/s, по которой планируется пропустить с разным качеством полос передачи данных до 300 клиентов, предполагается, что для системных нужд оставляется 1Gb/s, остальное нарезается некоторое число полос с дальнейшей утилизацией.
  + Предлагается ввести 3 вида качества предоставления услуг передачи данных (приоритеты) с нарезкой на 41 полосу:
    - высший, 26 клиентов с гарантированной шириной:
      * 2 полосы 1000Mb/s;
      * 4 полосы 200Mb/s;
      * 20 полос 100Mb/s;
    - средний, 70 клиентов с гарантированной утилизацией в 4 раза:
      * 4 полосы от 50 до 200Mb/s – 20 клиентов, гарантированная полоса 50Mb/s;
      * 10 полос от 20 до 100Mb/s – 50 клиентов, гарантированная полоса 20Mb/s;
    - низкий, до 204 клиентов с гарантированной полосой в 5Mb/s:
      * 1 полоса до 5Mb/s до 1000Mb/s;
      1. Описание формирования конфигурационных файлов с поддержкой выбранных методов управления трафиком в Cisco ASR, которым занимается системный программист.
  + Методов регулирования ширины полосы в семантике Cisco IOS конфигурации используется в данном случае два – т.н. police rate и shape average, что указывается в описании классов. Первый будет использован для низкоприоритетных клиентов, второй, более щадящий для TCP, для высшего и среднего.
  + Каждая политика состоит из 3-х описаний:
    - * ACL – список IP адресов клиентов
      * CLASS – класс политики, куда входит нужный ACL
      * POLICY-MAP – политика, применяемая к данному интерфейсу для входа и выхода и канальному интерфейсу транспортной сети
  + Для каждого из описаний в базе данных создается своя таблица с помощью следующих SQL–инструкций:

CREATE TABLE acls (

acl\_id integer DEFAULT nextval('acl\_id\_seq') NOT NULL,

acl\_name text NOT NULL,

ip inet NOT NULL,

rmask inet NOT NULL,

entry integer DEFAULT 0,

entry\_max integer DEFAULT 1,

cls\_id integer NOT NULL

);

CREATE TABLE classes (

cls\_id integer DEFAULT nextval('cls\_id\_seq') NOT NULL,

cls\_name text NOT NULL,

bandwidth integer NOT NULL,

priority integer,

pol\_id integer NOT NULL,

comment text

);

CREATE TABLE policies (

pol\_id integer DEFAULT nextval('pol\_id\_seq') NOT NULL,

pol\_name text NOT NULL,

comment text

);

* + В конфигурационном файле Cisco IOS необходимые строчки будут выглядеть следующим образом:

!

class-map match-all clsLim1r1Gb

match access-group name aclHi1r1Gb

class-map match-all clsLim2r1Gb

match access-group name aclHi2r1Gb

class-map match-all clsLim3r1Gb

match access-group name aclHi3r1Gb

class-map match-all clsLim4r1Gb

match access-group name aclHi4r1Gb

!

[..]

!

class-map match-all clsLim5Mb

match access-group name aclLow5Mb

!

policy-map hiPriority

class clsLim1r1Gb

shape average 1000000000

class clsLim2r1Gb

shape average 1000000000

class clsLim1r200Mb

shape average 200000000

class clsLim2r200Mb

shape average 200000000

class clsLim3r200Mb

shape average 200000000

class clsLim4r200Mb

shape average 200000000

class clsLim01r100Mb

shape average 100000000

class clsLim03r100Mb

shape average 100000000

class clsLim04r100Mb

shape average 100000000

class clsLim05r100Mb

shape average 100000000

class clsLim06r100Mb

shape average 100000000

class clsLim07r100Mb

shape average 100000000

class clsLim08r100Mb

shape average 100000000

class clsLim09r100Mb

shape average 100000000

class clsLim10r100Mb

shape average 100000000

class clsLim11r200Mb

shape average 200000000

class clsLim12r200Mb

shape average 200000000

class clsLim13r200Mb

shape average 200000000

class clsLim14r200Mb

shape average 200000000

class clsLim11r100Mb

shape average 100000000

class clsLim13r100Mb

shape average 100000000

class clsLim14r100Mb

shape average 100000000

class clsLim15r100Mb

shape average 100000000

class clsLim16r100Mb

shape average 100000000

class clsLim17r100Mb

shape average 100000000

class clsLim18r100Mb

shape average 100000000

class clsLim19r100Mb

shape average 100000000

class clsLim20r100Mb

shape average 100000000

!

policy-map midPriority

class clsLim1r50Mb

shape average 200000000

class clsLim2r50Mb

shape average 200000000

class clsLim3r50Mb

shape average 200000000

class clsLim4r50Mb

shape average 200000000

class clsLim01r20Mb

shape average 100000000

class clsLim02r20Mb

shape average 100000000

class clsLim03r20Mb

shape average 100000000

class clsLim04r20Mb

shape average 100000000

class clsLim05r20Mb

shape average 100000000

class clsLim06r20Mb

shape average 100000000

class clsLim07r20Mb

shape average 100000000

class clsLim08r20Mb

shape average 100000000

class clsLim09r20Mb

shape average 100000000

class clsLim10r20Mb

shape average 100000000

!

policy-map lowPriority

class clsLim5Mb

police rate 1000000000 bps

violate-action drop

1. Порядок и средства заполнения базы данных
   1. БД подсистемы мониторинга

В состав технических средств входит сервер ЦУМ. На каждом сервере ЦУМ запущена виртуальная машина «zabbix-N» (где N - номер ЦУМа), на которой установлены типовая база данных для Zabbix 5.0 «zabbix» на основе PostgreSQL 13, веб-интерфейс Zabbix и Zabbix сервер.

Заполнение базы данных мониторинга происходит в автоматическом режиме. Сбор данных осуществляется двумя способами:

1. Установленные на хостах Zabbix агенты собирают мониторинговую информацию и отправляют ее на оба Zabbix сервера (на каждый ЦУМ).
2. Zabbix сервер отправляет запросы на хосты и в ответе получает запрашиваемые данные.

Все полученные данные мониторинга Zabbix сервер записывает в таблицы «history», «history\_uint», «history\_str», «history\_log», «history\_text» базы данных «zabbix» в зависимости от типа данных.

Для отслеживания отклонения данных от приемлемых значений используются Zabbix триггеры. При срабатывании триггеров создаются события, которые записываются в таблицу «events» базы данных «zabbix».

Для каждого типа устройств создается свой шаблон Zabbix, который присоединяется к хостам (узлам сети). Все объекты (элементы данных, триггеры, графики и т.д.) из шаблона добавляются к этому хосту. Мониторинг выполняется в соответствии с указанными выше параметрами.

* 1. БД подсистемы технического учета

В состав технических средств входит сервер ЦУМ. На каждом сервере ЦУМ запущена виртуальная машина «glpi-N» (где N - номер ЦУМа), на которой установлены типовая база данных «glpi» для GLPI 9.5 на основе MariaDB 10 и веб-сервер GLPI. Для обнаружения состава оборудования и его загрузки в GLPI использовался плагин Fusioninventory.

Заполнение базы данных технического учета происходит в автоматическом и ручном режиме. Сбор данных осуществляется тремя способами:

1. Установленные на хостах Fusioninventory агенты собирают инвентарные данные и отправляют их на оба GLPI сервера (на каждый ЦУМ).
2. Администратор СУМ КС вручную добавляет техническую информацию через веб-интерфейс GLPI.
3. Модули подсистемы взаимодействия с внешними системами принимают описания из внешних систем и заносят их в базу подсистемы технического учета.

Все полученные данные инвентаризации GLPI сервер записывает в таблицы «glpi\_computers», «glpi\_networkequipments», «glpi\_racks», «glpi\_enclosures», «glpi\_pdus», «glpi\_passivedcequipments», «glpi\_networkports», «glpi\_softwares» базы данных «glpi» в зависимости от типа данных и оборудования.

* 1. Сервисные БД СУМ КС

В состав технических средств входит сервер ЦУМ. На каждом сервере ЦУМ запущена виртуальная машина «postgres-N» (где N - номер ЦУМа), на которой установлена СУБД PostgreSQL 13.

Заполнение сервисных баз данных выполняется по-разному в зависимости от сервиса, который использует эту БД.

Базы данных DNS и DHCP не требуют изменений в процессе эксплуатации. При модернизации или расширении всей системы системный программист вручную производит изменения в этих БД командами SQL, поддерживаемыми PostgreSQL.

* + 1. Таблицы БД ПО АРМ.

На каждом сервере ЦУМ запущена виртуальная машина «arm-N» (где N - номер ЦУМа), на которой установлено ПО АРМ, которое взаимодействует с базой данных «arm» на ВМ «postgres-N». Заполнение базы данных ПО АРМ происходит в автоматическом и ручном режиме. Сбор данных осуществляется двумя способами:

1. Администратор СУМ КС вручную вносит информацию о сеансах связи и пользователях через веб-интерфейс АРМ.
2. ПО АРМ запрашивает данные из других подсистем СУМ КС и заносит их в базу ПО АРМ.
   * 1. Таблицы БД подсистемы взаимодействия с внешними системами.

На каждом сервере ЦУМ запущена виртуальная машина «pum-outss-N» (где N - номер ЦУМа), на которой установлено ПО подсистемы взаимодействия с внешними системами, которое взаимодействует с базой данных «pum\_outss» на ВМ «postgres-N». Заполнение базы данных подсистемы взаимодействия с внешними системами происходит в автоматическом режиме. Сбор данных осуществляется двумя способами:

1. Модули подсистемы взаимодействия с внешними системами принимают данные из внешних систем и заносят их в базу соответствия объектов (таблицы «events\_table», «hosts\_table», «services\_table»).
2. Модули подсистемы взаимодействия с внешними системами запрашивают данные из других подсистем СУМ КС и заносит их в базу соответствия объектов (таблицы «events\_table», «hosts\_table», «services\_table»).
   * 1. Таблицы БД модуля управления ТТТ.
        1. Описание заполнения таблиц базы данных.

Заполнение таблиц acls, classes и policies выполняется согласно описанной в разделе 1.3.3 конфигурации либо вручную через панель администратора баз данных (модифицированная для пользователя), либо автоматически скриптом. В дальнейшем из данных таблиц формируется статическая конфигурация роутера Cisco ASR в части конфигурации политик и загружается единожды либо при форсировании конфигурации по командам администратора.

Динамическая конфигурация в части заполнения ACL и привязки конкретных политик к конкретным клиентским портам происходит при каждом запросе на создание или удаление клиентского подключения скриптами модуля управления трафиком. На данном этапе при появлении задания в очереди сообщений от RabbitMQ из АРМ создается дочерний процесс, запускающий серию скриптов, которые формируют задание на реконфигурацию в нужном ОУ соответствующего Cisco ASR по SNMP, по завершению реконфигурации этот же процесс заносит результат выполнения в ответную очередь сообщений, после чего он завершается.

Заполнение остальных таблиц МУ ТТТ происходит в автоматическом режиме при создании или выполнении запросов, а также путём реплицирования из базы АРМ согласно описанию таблиц БД МУ ТТТ из раздела 1.3.3.

* + - 1. Описание работы скриптов для заполнения таблиц базы данных.

Функционирование скриптов происходит в два этапа:

* + - Реплицируются данные из базы данных АРМ по соответствующим view в таблицы «ports» и «switches» по установленным флагам актуальности в таблице базы АРМ.
    - Заполняются таблицы конфигурации по параметрам и файлам из командной строки.

1. Процедуры изменения и контроля базы данных
   1. БД подсистемы мониторинга

Детальные сведения о составе и последовательности выполнения процедур по изменению содержания базы данных изложены в разделах «Правила подготовки данных» и «Порядок и средства заполнения базы данных».

Прямой доступ администраторов к базе данных не предполагается. Ввод и корректировка данных осуществляется только с помощью программных компонент СУМ КС. Контроль программной корректности содержания базы данных производится заимствованной системой Zabbix, лежащей в основе подсистемы мониторинга.

* 1. БД подсистемы технического учета

Детальные сведения о составе и последовательности выполнения процедур по изменению содержания базы данных изложены в разделах «Правила подготовки данных» и «Порядок и средства заполнения базы данных».

Прямой доступ администраторов к базе данных не предполагается. Ввод и корректировка данных осуществляется с помощью программных компонент СУМ КС. Контроль программной корректности содержания базы данных производится заимствованной системой GLPI, лежащей в основе подсистемы технического учета.

* 1. Сервисные БД СУМ КС

Детальные сведения о составе и последовательности выполнения процедур по изменению содержания базы данных изложены в разделах «Правила подготовки данных» и «Порядок и средства заполнения базы данных».

Прямой доступ администраторов к базе данных не предполагается. Ввод и корректировка данных осуществляется с помощью программных компонент СУМ КС. Контроль программной корректности содержания базы данных производится сервисами, которые используют требуемую БД.

1. Порядок и средства восстановления базы данных
   1. БД подсистемы мониторинга
      1. Основными средствами защиты данных БД подсистемы мониторинга являются системные средства аутентификации и шифрования СУБД PostgreSQL.

В веб-интерфейсе каждый пользователь имеет доступ на чтение и запись только строго определенного набора информации в соответствии со своей ролью.

Период хранения исторических данных в БД – 3 года.

* + 1. Операция резервирования и восстановления базы данных включает в себя:

1. Создание резервной копии БД;
2. Архивацию резервных копий для экономии дискового пространства;
3. Автоматизацию выполнения операции резервирования путем настройки планировщика операционной системы;
4. Восстановление БД из резервной копии.
   * 1. Краткое описание организации резервного копирования данных подсистемы мониторинга:
5. ВМ «zabbix-N» снабжена скриптом (/usr/local/bin/backup-zabbix.sh) для создания резервной копии в виде архива. Локальные резервные копии хранятся по пути «/home/backup/zabbix\_backup».
6. Сервер ЦУМ снабжен скриптом (/usr/local/bin/backup\_zabbix\_file.sh) для удаленного подключения к ВМ «zabbix-N», запуска скрипта резервного копирования под пользователем backup и отправки резервной копии на другой сервер ЦУМ. Резервные копии сохраняются на удаленный сервер ЦУМ по пути «srv/backup/zabbix». Лог пишется в системный журнал syslog.
7. Cкрипт на сервере ЦУМ запускается сервисом «backup\_zabbix\_file.service». Этот сервис в свою очередь запускается системным таймером «backup\_zabbix\_file.timer» ежедневно ночью. Администратору не требуется выполнять скрипты резервного копирования вручную.
8. Подключение к ВМ «zabbix-N» и на удаленный сервер ЦУМ выполняется по ключу. Закрытый ключ хранится на каждом ЦУМ (/home/backup/.ssh/id\_rsa). Данные открытого ключа передаются на сервера ЦУМ и ВМ «zabbix-N» для пользователя «backup».
   * 1. Краткое описание организации восстановления из резервной копии данных подсистемы мониторинга:
9. ВМ «zabbix-N» снабжена скриптом (/usr/local/bin/restoring\_dump-zabbix.sh) для восстановления БД из резервной копии. На вход скрипта подается полный путь архива резервной копии. Скрипт выполняет разархивирование дампа базы данных, останавливает работу сервиса, выполняет восстановление БД, запускает сервис.
10. Сервер ЦУМ снабжен скриптом (/usr/local/bin/restoring\_dump\_zabbix\_file.sh) для удаленного подключения к ВМ «zabbix-N», скачивания необходимой резервной копии с сервера хранения резервных копий (с другого сервера ЦУМ) и запуска скрипта восстановления БД из скачанной резервной копии. Лог пишется в системный журнал syslog.
11. Скрипт запускается вручную администратором при необходимости произвести восстановление из резервной копии. Для скрипта вводится аргумент в виде полного пути необходимой резервной копии. Затем администратору требуется подтвердить операцию восстановления и выбранную версию резервной копии.
12. Подключение к ВМ «zabbix-N» и на удаленный сервер ЦУМ выполняется по ключу. Закрытый ключ хранится на каждом ЦУМ (/home/backup/.ssh/id\_rsa). Данные открытого ключа передаются на сервера ЦУМ и ВМ «zabbix-N» для пользователя «backup».
    1. БД подсистемы технического учета
       1. Основными средствами защиты данных БД подсистемы технического учета являются системные средства аутентификации и шифрования СУБД MariaDB.

В веб-интерфейсе каждый пользователь имеет доступ на чтение и запись только строго определенного набора информации в соответствии со своей ролью.

Период хранения исторических данных в БД – 3 года.

* + 1. Операция резервирования и восстановления базы данных включает в себя:

1. Создание резервной копии БД;
2. Архивацию резервных копий для экономии дискового пространства;
3. Автоматизацию выполнения операции резервирования путем настройки планировщика операционной системы;
4. Восстановление БД из резервной копии.
   * 1. Краткое описание организации резервного копирования данных подсистемы технического учета:
5. ВМ «glpi-N» снабжена скриптом (/usr/local/bin/backup-glpi.sh) для создания резервной копии в виде архива. Локальные резервные копии хранятся по пути «/home/backup/glpi\_backup».
6. Сервер ЦУМ снабжен скриптом (/usr/local/bin/backup\_glpi\_file.sh) для удаленного подключения к ВМ «glpi-N», запуска скрипта резервного копирования под пользователем backup и отправки резервной копии на другой сервер ЦУМ. Резервные копии сохраняются на удаленный сервер ЦУМ по пути «srv/backup/glpi». Лог пишется в системный журнал syslog.
7. Cкрипт на сервере ЦУМ запускается сервисом «backup\_glpi\_file.service». Этот сервис в свою очередь запускается системным таймером «backup\_glpi\_file.timer» ежедневно ночью. Администратору не требуется выполнять скрипты резервного копирования вручную.
8. Подключение к ВМ «glpi-N» и на удаленный сервер ЦУМ выполняется по ключу. Закрытый ключ хранится на каждом ЦУМ (/home/backup/.ssh/id\_rsa). Данные открытого ключа передаются на сервера ЦУМ и ВМ «glpi-N» для пользователя «backup».
   * 1. Краткое описание организации восстановления из резервной копии данных подсистемы мониторинга:
9. ВМ «glpi-N» снабжена скриптом (/usr/local/bin/restoring\_dump-glpi.sh) для восстановления БД из резервной копии. На вход скрипта подается полный путь архива резервной копии. Скрипт выполняет разархивирование дампа базы данных, останавливает работу сервиса, выполняет восстановление БД, запускает сервис.
10. Сервер ЦУМ снабжен скриптом (/usr/local/bin/restoring\_dump\_glpi\_file.sh) для удаленного подключения к ВМ «glpi-N», скачивания необходимой резервной копии с сервера хранения резервных копий (с другого сервера ЦУМ) и запуска скрипта восстановления БД из скачанной резервной копии. Лог пишется в системный журнал syslog.
11. Скрипт запускается вручную администратором при необходимости произвести восстановление из резервной копии. Для скрипта вводится аргумент в виде полного пути необходимой резервной копии. Затем администратору требуется подтвердить операцию восстановления и выбранную версию резервной копии.
12. Подключение к ВМ «glpi-N» и на удаленный сервер ЦУМ выполняется по ключу. Закрытый ключ хранится на каждом ЦУМ (/home/backup/.ssh/id\_rsa). Данные открытого ключа передаются на сервера ЦУМ и ВМ «glpi-N» для пользователя «backup».
    1. Сервисные БД СУМ КС

Основными средствами защиты данных сервисных БД СУМ КС являются системные средства аутентификации и шифрования СУБД PostgreSQL.

В веб-интерфейсе АРМ каждый пользователь имеет доступ на чтение и запись только строго определенного набора информации в соответствии со своей ролью.

С помощью Bucardo между обоими ЦУМ настроена репликация таблиц PostgreSQL типа master <-> master. Такая репликация позволяет хранить резервные копии на другом ЦУМ. При восстановлении связи между ЦУМ после аварии происходит автоматическое восстановление данных из другого ЦУМ.

Перечень принятых сокращений

|  |  |
| --- | --- |
| ACL | Список IP адресов клиентов |
| CAS | Central Authentication Service |
| CWDM | Coarse Wavelength Division Multiplexing |
| DHCP | Dynamic Host Configuration Protocol |
| DNS | Domain Name System |
| HTTP | HyperText Transfer Protocol |
| ID | Identifier |
| IOS | Internetwork Operating System |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| LDAP | Lightweight Directory Access Protocol |
| SNMP | Simple Network Management Protocol |
| SQL | Structured Query Language |
| TCP | Transmission Control Protocol |
| URL | Uniform Resource Locator |
| VLAN | Virtual Local Area Network |
| АРМ | Автоматизированное рабочее место |
| БД | База данных |
| ВМ | Виртуальная машина |
| ИБП | Источник бесперебойного питания |
| ККП | Квантовая коммуникационная платформа цифровой экономики |
| МУ ТТТ | Модуль управления трафиком, топологией и технологическими параметрами квантовой коммуникационной платформы |
| ОУ | Опорный узел |
| ОУТСС | Система оперативного контроля и управления технологическими сетями связи |
| «ФПСУ-IP» 10G | Подсистема организации доверенных опорных узлов квантовой связи |
| ПО | Программное обеспечение |
| ПОУ | Промежуточный опорный узел |
| ПУ МКС | Пилотный участок магистральной квантовой сети |
| СУМ КС | Подсистема управления и мониторинга квантовой сетью |
| СУБД | Система управления базами данных |
| УК | Учетная карточка |
| ЦУМ | Центр управления и мониторинга |