UNLIMITED PRODUCTION

eXpress

Communication System

Microsoft Outlook Add-in

Installation Guide

Keycloak Version 3.1.0.2

Mailing address:	127030, Moscow, 24/1 Novoslobodskaya Street,
Phone:	+7 (499) 288-01-22
E-mail:	sales@express.ms
Web:	https://express.ms/

TABLE OF CONTENTS

TERMS AND DEFINITIONS

Term	**Definition**
AD	Active Directory — Microsoft's directory service for the Windows Server operating system family
API	Application Programming Interface — an interface for interaction between programs and applications
Chat Bot	Conference Notifier Bot chat bot
Conference Creator	The user who creates and can edit an Express conference using the Outlook add-in
Container	A standardized, isolated, and portable software package that includes everything needed to run an application, including code, runtime, system tools, libraries, and settings.
CTS	Corporate Transport Server
DB	Database
DBMS	Database Management System
Docker	An open-source containerization platform that automates application creation, delivery, and management
Docker-compose	An add-on (utility) over Docker, a Python application that allows running multiple containers simultaneously and routing data flows between them.
ETS	Enterprise Transport Server
express-core-service	A server application for the Outlook add-in that provides core functionality (creating and modifying conferences)
express-template-service	A server application for the Outlook add-in that provides additional functionality (creating and modifying email templates, text, logo, signature, etc.)
Express CS, Express, System	Express Communication System
Invited Participant	User(s) invited by the Express conference creator using the Outlook add-in
JSON	A text-based data exchange format based on JavaScript
KeyCloak	An application for implementing a single point of identification, authentication, and authorization. It is an open-source identity and access management system
Log	An entry in the server and/or client event log
MS Exchange	Microsoft Mail Server
MS Outlook	Microsoft mail client, part of the MS Office application package
OS	Operating System
PC	Personal Computer
Single CTS	Unified Corporate Transport Server
Split CTS	Split Corporate Transport Server: Front CTS and Back CTS
SW	Software

Chapter 1

GENERAL REQUIREMENTS

Attention! The following competencies are required to perform the operations in this instruction:

- Windows Server administration;
- Keycloak administration;
- PostgreSQL administration;
- Linux administration;
- Docker administration;
- eXpress administration;
- understanding of JSON.

Before starting, ensure that software and hardware meet the following requirements:

- Docker version not lower than Docker version 24.0.7;
- Docker-compose version not lower than version 1.29.2;
- Database requirements:
  - PostgreSQL version not lower than (Debian 13.11-0+deb11u1, Windows);
  - superuser privileges on the PostgreSQL server;
  - a dedicated PostgreSQL database and a dedicated PostgreSQL database user: this user must be the owner of this database and have all privileges on it;
- availability of an installed and configured Keycloak solution, version not lower than 22.0.1;
- Outlook client version not lower than 2013;
- Outlook must be used with a mailbox that corresponds to the email address registered in the Express CS account;
- availability of an installed and configured Express CS solution, version not lower than 3.8;
- the Conference Notifier Bot must be configured in Express CS. Requirements for chatbot configuration and methods for checking correctness are described in Chapter 3 "Installing and Configuring Conference Notifier Chat Bot";
- network interaction between components must be configured on the appropriate ports;
- WebView2 Runtime package, version not lower than 122.0.2365.66 (https://developer.microsoft.com/ru-ru/microsoft-edge/webview2/?form=MA13LH), must be installed on client PCs.

If there are no other services on the virtual machine, the Docker server must meet the following technical requirements (Table 1):

Table 1

Component	**Parameters**
Processor	64-bit 2.4 GHz processor 2 core
RAM	4 GB
Operating System	Debian GNU/Linux 11 or later
Hard Disk	At least 40 GB

The PostgreSQL DBMS can be located on a dedicated server running Windows OS or Linux OS. If the pre-installed PostgreSQL DBMS is absent, and the architecture of the deployed solution assumes the presence of an Express CS Bot server, a PostgreSQL docker container (which is part of the Bot-server) can be used to create the DB. A Linux server, where the Outlook add-in will be deployed, can be used to create the PostgreSQL DB using the PostgreSQL docker container.

Important! It is prohibited to use the following Express CS servers for creating the PostgreSQL database: CTS Back, Front, Media, Rec, ETS, and others (except for the Bot server).

If the virtual machine will not host other services, the PostgreSQL DBMS server for Windows OS must meet the following technical requirements (Table 2):

Table 2

Component	Parameters
Processor	64-bit 1.4 GHz processor 2 core
RAM	2 GB
Operating System	Debian GNU/Linux 11 or later
Hard Disk	At least 40 GB

If the PostgreSQL DBMS is additionally deployed on the Linux OS server for the Outlook add-in, add 2 GB RAM and 20 GB HDD.

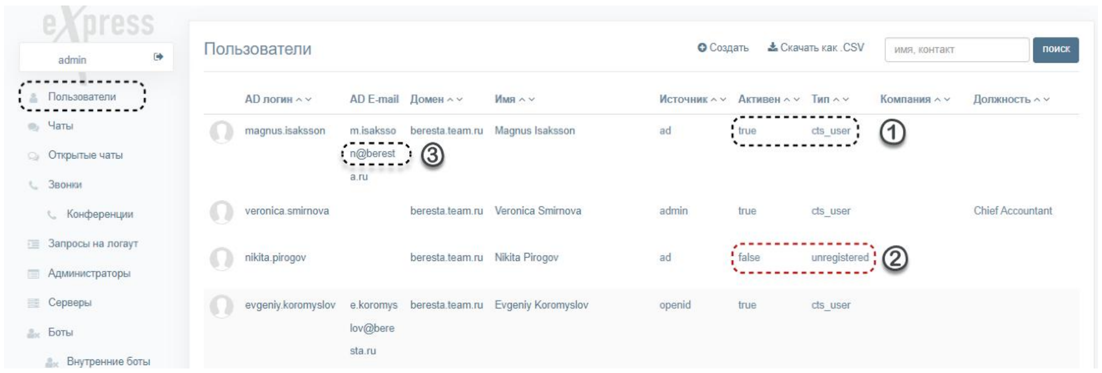
If there are no other services on the virtual machine, the PostgreSQL DBMS server under Windows OS must meet the following technical requirements (Table 3):

Table 3

Component	Parameters
Processor	64-bit 1.4 GHz processor 2 core
RAM	2 GB
Operating System	Windows Server 2016 or later
Hard Disk	At least 40 GB

All users who want to use the add-in must be registered in the Express CS and be active. The user status can be checked in the Express CS administrator web interface – "Users" section (Figure 1).

Important! When using MS Exchange, the conference creator must be registered in the Express CS with the email address specified as their Primary SMTP address in MS Exchange.



AD логин	AD E-mail	Домен	Имя	Источник	Активен	Тип	Компания	Должность
magnus.isaksson	m.isakssо n@berest a.ru	beresta.team.ru	Magnus Isaksson	ad	true	cts_user		
veronica.smirnova		beresta.team.ru	Veronica Smirnova	admin	true	cts_user		Chief Accountant
nikita.pirogov		beresta.team.ru	Nikita Pirogov	ad	false	unregistered		
evgeniy.koromyslov	e.koromys lov@bere sta.ru	beresta.team.ru	Evgeniy Koromyslov	openid	true	cts_user		

Figure 1. User Status: 1 – Active; 2 – Inactive

For an upcoming conference to be automatically added in the Express CS application of the invited user, that user must be registered in the application with the same email address that the conference creator used to send the invitation via the add-in (Figure 1, indicator 3).

The user will receive an email with a direct join link if:

- the user is not present in the Express CS;
- the user has an "Inactive" status in the Express CS;
- the user is registered in the Express CS with a different email address.

Chapter 2

ARCHITECTURE

Different deployment schemes for the Outlook add-in server-side are used depending on the Express CS architectural solution employed. The most common (standard) architectural solutions are described below.

Specific deployment schemes are determined by customer requirements after consultation with the development company.

If a single email template is planned for use, all express-core-service instances can use one database. If different templates are required for different express-core-service instances, then a separate database and a separate express-template-service should be deployed for each of them.

SINGLE CTS OR SPLIT CTS

If Express CS contains a single CTS server, deploy the server part of the add-in in a single instance. The typical deployment scheme for this type is presented below (Figure 2):

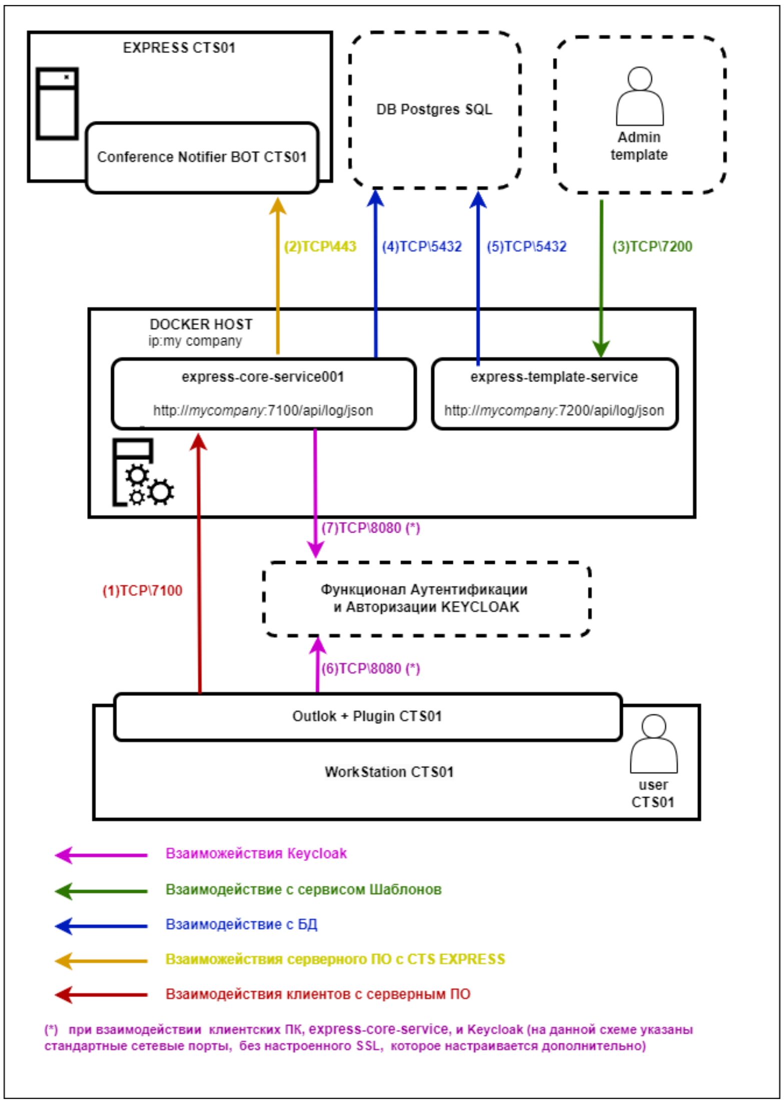

EXPRESS CTS01
Conference Notifier BOT CTS01
DB Postgres SQL
Admin template
(2)TCP\443
(4)TCP\5432
(5)TCP\5432
(3)TCP\7200
DOCKER HOST
ip:my company
express-core-service001
http://mycompany:7100/api/log/json
express-template-service
http://mycompany:7200/api/log/json
(7)TCP\8080 (*)
(1)TCP\7100
Функционал Аутентификации
и Авторизации KEYCLOAK
(6)TCP\8080 (*)
Outlok + Plugin CTS01
WorkStation CTS01
user CTS01
Взаиможействия Keycloak
Взаимодействие с сервисом Шаблонов
Взаимодействие с БД
Взаиможействия серверного ПО с CTS EXPRESS
Взаимодействия клиентов с серверным ПО
(*) при взаимодействии клиентских ПК, express-core-service, и Keycloak (на данной схеме указаны стандартные сетевые порты, без настроенного SSL, которое настраивается дополнительно)

Figure 2. Typical Scheme: 1 CTS, 1 express-core-service, 1 express-template-service

The network interaction numbers correspond to the row number in Table 4 Appendix 3.

MULTIPLE CTS

If Express CS contains multiple CTS servers, deploy the server part of the add-in separately for each CTS server (deployment on a single Docker server, but with different ports and container names, or on multiple Docker servers is allowed). The typical deployment scheme for this type is presented below (Figure 3):

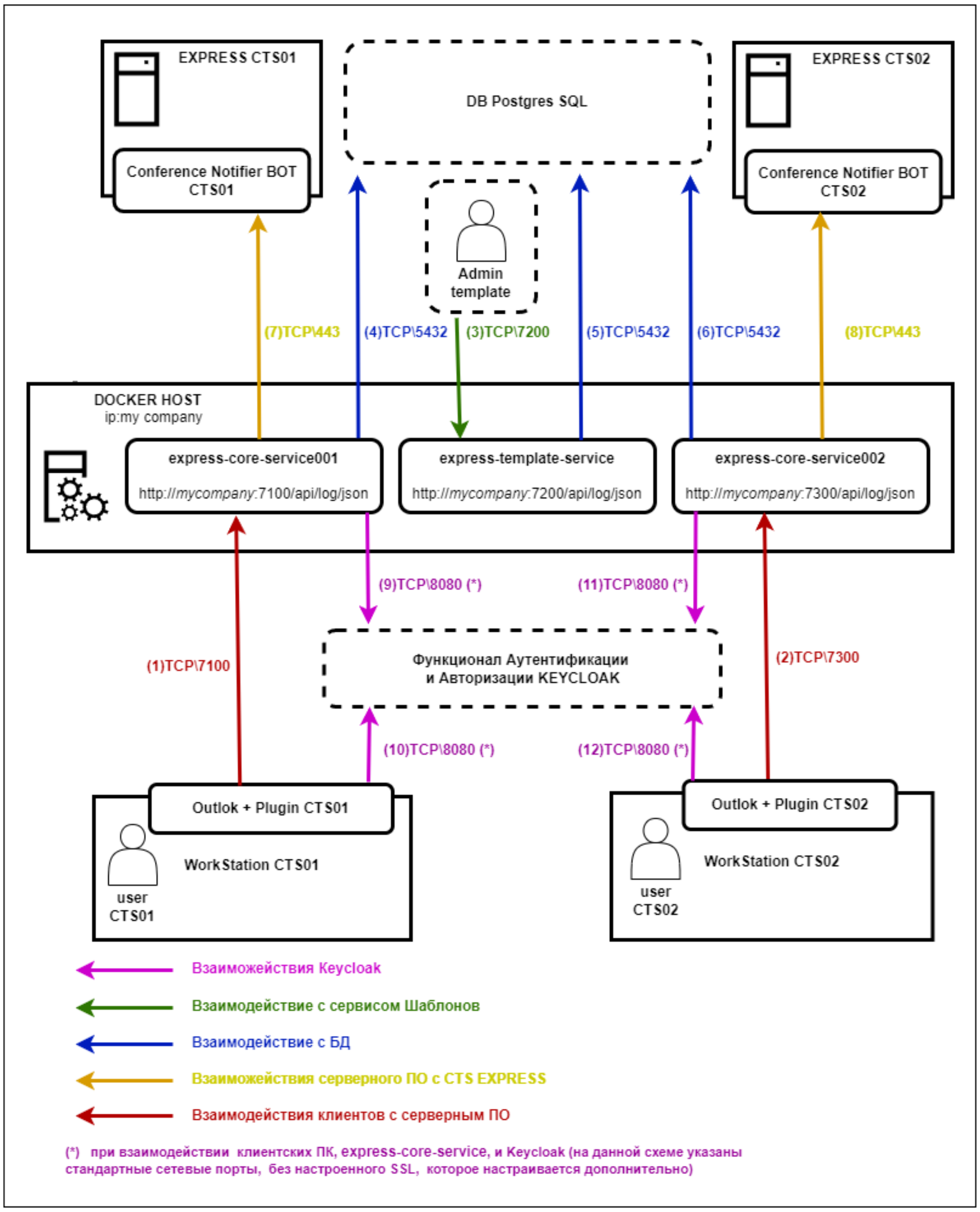

EXPRESS CTS01
Conference Notifier BOT CTS01
DB Postgres SQL
EXPRESS CTS02
Conference Notifier BOT CTS02
Admin template
(7)TCP\443
(4)TCP\5432
(3)TCP\7200
(5)TCP\5432
(6)TCP\5432
(8)TCP\443
DOCKER HOST
ip:my company
express-core-service001
http://mycompany:7100/api/log/json
express-template-service
http://mycompany:7200/api/log/json
express-core-service002
http://mycompany:7300/api/log/json
(9)TCP\8080 (*)
(11)TCP\8080 (*)
(1)TCP\7100
Функционал Аутентификации и Авторизации KEYCLOAK
(2)TCP\7300
(10)TCP\8080 (*)
(12)TCP\8080 (*)
Outlok + Plugin CTS01
user CTS01
WorkStation CTS01
Outlok + Plugin CTS02
user CTS02
WorkStation CTS02
Взаможействия Keycloak
Взаимодействие с сервисом Шаблонов
Взаимодействие с БД
Взаможействия серверного ПО с CTS EXPRESS
Взаимодействия клиентских с серверным ПО
(*) при взаимодействии клиентских ПК, express-core-service, и Keycloak (на данной схеме указаны стандартные сетевые порты, без настроенного SSL, которое настраивается дополнительно)

Figure 3. Typical Scheme: 2 CTS, 2 express-core-service, 1 express-template-service

The network interaction numbers correspond to the row number in Table 5 Appendix 3.

MULTIPLE CTS AND ETS

If Express CS contains multiple CTS servers unified by an ETS server, deploy the server part of the add-in only for the ETS server. The typical deployment scheme for this type is presented below (Figure 4):

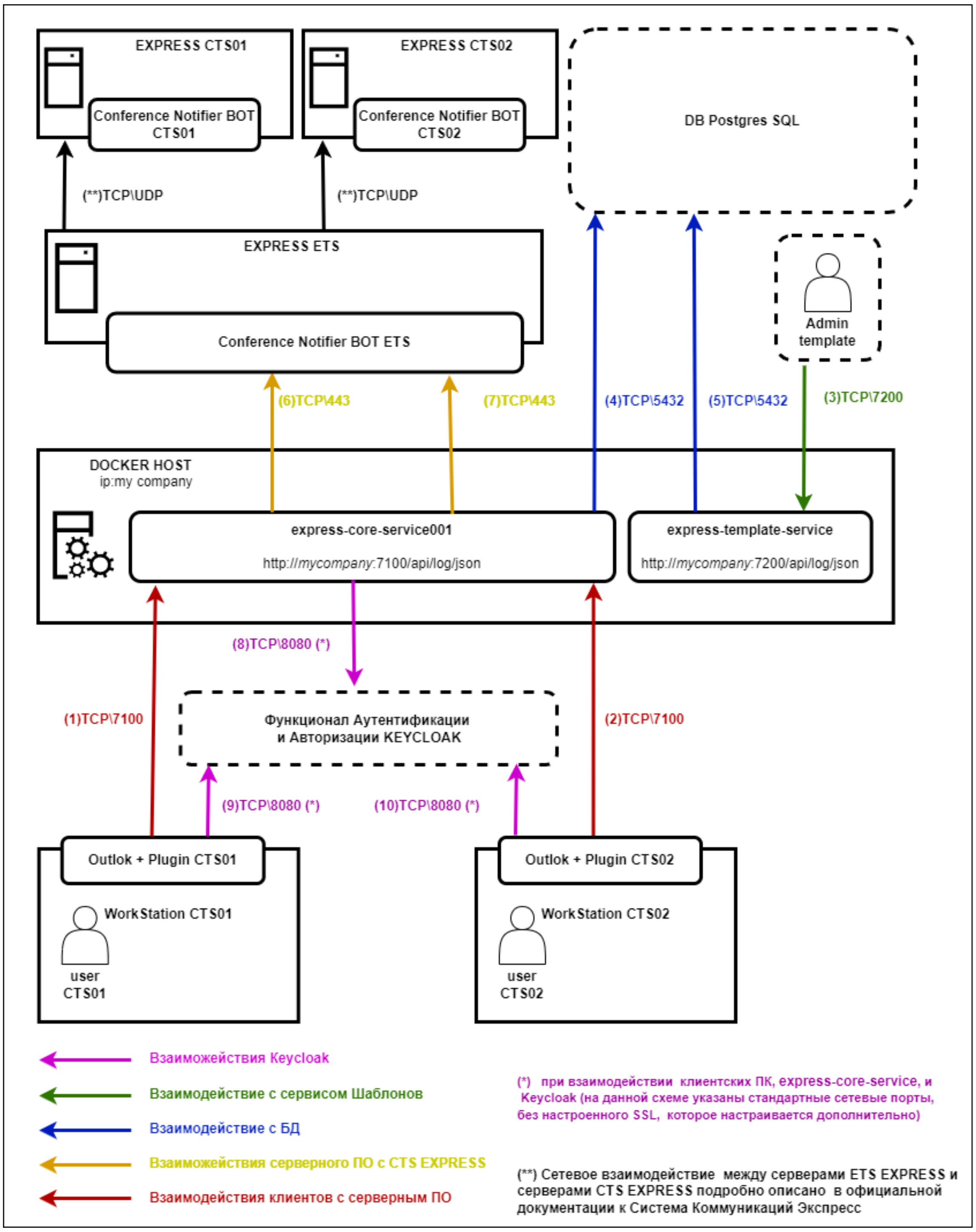

EXPRESS CTS01
Conference Notifier BOT CTS01
EXPRESS CTS02
Conference Notifier BOT CTS02
DB Postgres SQL
(**)TCP\UDP
(**)TCP\UDP
EXPRESS ETS
Conference Notifier BOT ETS
Admin template
(6)TCP\443
(7)TCP\443
(4)TCP\5432
(5)TCP\5432
(3)TCP\7200
DOCKER HOST
ip:my company
express-core-service001
http://mycompany:7100/api/log/json
express-template-service
http://mycompany:7200/api/log/json
(8)TCP\8080 (*)
(1)TCP\7100
Функционал Аутентификации и Авторизации KEYCLOAK
(2)TCP\7100
(9)TCP\8080 (*)
(10)TCP\8080 (*)
Outlook + Plugin CTS01
WorkStation CTS01
user CTS01
Outlook + Plugin CTS02
WorkStation CTS02
user CTS02
Взаиможействия Keycloak
Взаимодействие с сервисом Шаблонов
Взаимодействие с БД
Взаиможействия серверного ПО с CTS EXPRESS
Взаимодействия клиентов с серверным ПО
(*) при взаимодействии клиентских ПК, express-core-service, и Keycloak (на данной схеме указаны стандартные сетевые порты, без настроенного SSL, которое настраивается дополнительно)
(**) Сетевое взаимодействие между серверами ETS EXPRESS и серверами CTS EXPRESS подробно описано в официальной документации к Система Коммуникаций Экспресс

Figure 4. Typical Scheme: 1 ETS, 2 CTS, 1 express-core-service, 1 express-template-service

The network interaction numbers correspond to the row number in Table 6 Appendix 3.

Chapter 3

INSTALLING AND CONFIGURING CONFERENCE NOTIFIER CHAT BOT

This section describes the procedure for connecting the Conference Notifier Bot chat bot API on the CTS server of the Express CS software solution.

Note. The procedure is described for Debian 11.4 and Ubuntu 2X. If issues arise while configuring the chat bot on other operating systems, it is recommended to contact the development company.

STEP 1. ENABLING CONFERENCE NOTIFIER BOT API WITHOUT PASSWORD

Important! When configuring the chat bot without protection, any user from the external network segment can use it if it is open.

Key points:

- if the Express CS contains multiple CTS servers, this procedure must be performed on each CTS that will interact with express-core-service;
- If the Express CS contains one or multiple ETS servers, this procedure must be performed on each ETS and CTS that will interact with express-core-service.

To enable the Conference Notifier Bot API without a password:

1. On the CTS server, add the following code to the settings.yaml[1] file:

```
conference_bot_env_override:
  API_ENABLED: true
```

   In a split-server architecture, changes are made only to the file on the Back CTS server.

2. Save the file and run from /OPT/EXPRESS:

```
dpl -d conference_bot
```

3. Open the administrator web interface and navigate to "Bots → Internal Bots → Conference Notifier Bot".

   The "Edit Bot" window will open.

4. In the opened window, select the checkbox in the "Enabled" field and click the "Save" button. Add port 4000 if necessary.

5. Check the chat bot's accessibility using a CURL request, for example:

```
curl -X POST -H "Content-Type: application/json" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

   Note. Parameters marked in red must be updated according to the specific request.

   If the chat bot is accessible, CURL will output:

```
"status":"ok"
```

6. In the CTS server administrator web interface, navigate to "Calls → Conferences".

7. In the search field, enter test_plugin_001 (specified in step 5) and click "Search".

   If the table displays a conference named test_plugin_001, the chat bot is configured correctly.

[1] The presumed path is /opt/express/settings.yaml.

To check the Docker container status after installing the bot on the Express server, enter the command:

```
docker ps
```

The conference_bot container should have the statuses "Up" and "healthy".

Important! The versions of the conference_bot and messaging containers must match. If they do not match, the entire Express CS should be updated.

STEP 2. SECURING CONFERENCE NOTIFIER BOT

Key points:

- if the Express CS contains multiple CTS servers, this procedure must be performed on each CTS that will interact with express-core-service;
- if the Express CS contains a single unified ETS, this procedure should be performed only on it.

To secure the Conference Notifier Bot, perform the following bearer token setup:

1. Ensure that Express CS version 3.8 or later is installed. If the version is lower, update Express to the latest version.
2. Ensure that the distribution version of the "Express CS Add-in for Outlook" (server and client) is 1.2.0.0 or later.

   Attention! The entered credentials will be used when configuring the Outlook add-in.

3. Generate a BEARER TOKEN of 30 to 40 characters using a password generation tool or by using the following commands:

   ```
   openssl rand -hex 40
   ```

   or

   ```
   head /dev/urandom | tr -dc A-Za-z0-9 | head -c 40 ; echo
   ```

4. On the CTS server, add the following code to the conference_bot_env_override: section in the settings.yaml[2] file:

   ```
     API_AUTH_METHOD: BEARER_TOKEN
     BEARER_TOKEN: 3bDewf52b3268sdg59f1f7fff33w01dd3c0431
   ```

   The conference_bot_env_override section should then look like this:

   ```
   conference_bot_env_override:
     API_ENABLED: true
     API_AUTH_METHOD: BEARER_TOKEN
     BEARER_TOKEN: 3bDewf52b3268sdg59f1f7fff33w01dd3c0431
   ```

   In the case of a split server architecture, changes are made only to the file on the Back CTS server.

   The token value should be invented by you or generated by any third-party software (it must be at least 40 characters long and contain uppercase and lowercase Latin letters and numbers).

5. Save the file and run the following command from /OPT/EXPRESS:

   ```
   dpl -d conference_bot
   ```

6. Check the unavailability of the chatbot with a request without a bearer token using CURL, for example:

[2] The presumed path is /opt/express/settings.yaml

```
curl -X POST -H "Content-Type: application/json" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

Note. Parameters marked in red must be updated according to the specific request.

If the chatbot is unavailable without a bearer token (which is expected), CURL will output an empty string.

7. Check the availability of the chatbot with a request with a bearer token using CURL, for example:

```
curl -X POST -H "Content-Type: application/json" -H "Authorization:
Bearer 3bfef52b32685srdsrhderhFGd1f73301dd3c0431" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

Note. Parameters marked in red must be updated according to the specific request.

If the chatbot is available, CURL will output:

```
"status":"ok"
```

8. In the administrator web interface of the CTS server, go to the “Calls Conferences” section.

9. In the search field, enter test_plugin_001 (set in item 6) and click the “Search” button.

   If a conference with the name test_plugin_001 is displayed in the table, the chatbot is configured correctly.

Chapter 4

INSTALLING THE OUTLOOK ADD-IN

To install the Outlook add-in, perform the steps described below. The distribution (server-side and client-side) for download is available at the following link: https://nc.express.ms/s/f3iGiJieaGyPM7j?path=%2FKeycloak (select the current product version).

This step is performed if the service has already been deployed as part of the previous add-in version.

STEP 1. UPDATING THE SERVER PART (OPTIONAL)

To update the server part:

1. Download the new versions of the express-meeting-core-service and express-meeting-template-service images to Docker.
2. In the folders containing the ENV and YML files, run the command:
```
docker-compose --env-file ./.env -f ./docker-compose.yml down
```
3. Take the new ENV and YML files from the new distribution version and specify your parameters and new image versions in them.
4. Replace the old ENV and YML files with the new ones in the corresponding folder on the Docker server.
5. Perform "Step 2. Preparing the Database" of the current chapter to update the DB structure.
6. Start the containers from the docker folder:
```
docker-compose --env-file ./.env -f ./docker-compose.yml up -d
```

STEP 2. PREPARING THE DATABASE

To deploy the DB that stores the invitation block templates used when creating an invitation in an Outlook email:

1. Go to the Windows machine that has access to the PostgreSQL server. The machine must have .NET Desktop Runtime 7.0 (https://dotnet.microsoft.com/en-us/download/dotnet/7.0).
2. Transfer the "tools" folder from the distribution package to the machine.
3. In the tools\appsettings.json file, set the connection string for the PostgreSQL DB in the key:
```
"Connection":"Host=127.0.0.1;Port=5432;Database=express_meeting_db;U
sername=db_express;Password=Pass1234;Pooling=true;Minimum Pool
Size=50;Maximum Pool Size=100;Include Error Detail=True;"
```
Note. The parameters Username=db_express and Password=Pass1234 must correspond to the credentials that were created during PostgreSQL configuration.

   Save the changes.
4. Run the following from the "tools" folder in the console:
```
ExpressMeeting.Tools.TemplatesDB.exe --ef-migrate
```
5. Wait for the utility to complete its work.

   The DB is ready for use.

STEP 3. DEPLOYING EXPRESS-CORE-SERVICE AND TEMPLATE-CORE-SERVICE IN DOCKER

The service is deployed in Docker.

To deploy the service:

1. Load the service image files into Docker from the images folder:

   ```
   docker load --input express-meeting-core-service-main.tar
   docker load --input express-meeting-template-service-main.tar
   ```

2. Prepare the service startup settings in the docker\.env file.

   The file contains the following settings:

   - the name of the image from which the core-service container starts (do not change):

     ```
     EXPRESS_MEETING_CORE_SERVICE_MAIN_IMAGE_NAME=express-meeting-
     core-service-main
     ```

   - the port through which core-service will be accessible:

     ```
     EXPRESS_MEETING_CORE_SERVICE_PORT=7100
     ```

   - the name of the image from which the template-service container starts (do not change):

     ```
     EXPRESS_MEETING_TEMPLATE_SERVICE_MAIN_IMAGE_NAME=express-meeting-
     template-service-main
     ```

   - the port through which template-service will be accessible:

     ```
     EXPRESS_MEETING_TEMPLATE_SERVICE_PORT=7200
     ```

   - the IP address corresponding to the DNS name of the server where the Conference Notifier Bot API is deployed (specify actual data):

     ```
     EXPRESS_MEETING_EXTRA_HOST=api.bot.express:127.0.0.1
     ```

   - the IP address corresponding to the DNS name of the Keycloak server (specify actual data):

     ```
     EXPRESS_MEETING_EXTRA_HOST2=keycloak.mycompany:127.0.0.1
     ```

   - the name of the file with core-service settings (do not change):

     ```
     EXPRESS_MEETING_CORE_ENV_FILE=core.env
     ```

   - the name of the file with template-service settings (do not change):

     ```
     EXPRESS_MEETING_TEMPLATE_ENV_FILE=template.env
     ```

   - the container folder for service logs (do not change):

     ```
     EXPRESS_MEETING_DOCKER_LOGS_PATH=/public/express-meeting-logs
     ```

   - the Docker host folder where service logs will be added from the containers (change to the required path if necessary):

     ```
     EXPRESS_MEETING_LOGS_PATH=/public/express-outlook/logs
     ```

3. Prepare the core-service startup settings in the docker\core.env file.

   The file contains the following settings:

   - the address of the Conference Notifier Bot API (specify the actual address where the API is deployed):

     ```
     ExpressOptions__Uri=https://api.bot.express/api/v1/conference_bot
     /conferences/
     ```

   - the authorization scheme for the Conference Notifier Bot API (Accepted values: Basic – login/password authorization; StaticToken – static token authorization):

     ```
     ExpressOptions__AuthenticationScheme=StaticToken
     ```

   - the token that will be used when accessing the Conference Notifier Bot API (used if the ExpressOptions__AuthenticationScheme parameter is set to StaticToken):

     ```
     ExpressOptions__Token=Bearer 3bb326abc5609fed301dd3c0431
     ```

   - the login of the account used to access the Conference Notifier Bot API (used if the ExpressOptions__AuthenticationScheme parameter is set to Basic):

```
ExpressOptions__UserLogin=admin
```

- the password of the account used to access the Conference Notifier Bot API (used if the ExpressOptions__AuthenticationScheme parameter is set to Basic):
```
ExpressOptions__UserPassword=admin
```
- the authorization type for core-service (do not change):
```
AuthorizationOptions__AuthType=keycloak
```
- the Keycloak address (specify actual):
```
AuthorizationOptions__Configuration__Url=https://keycloak.mycompa
ny
```
- the realm used for authorization in Keycloak (specify actual):
```
AuthorizationOptions__Configuration__Realm=realm
```
- the Client ID value used for authorization in Keycloak (specify actual):
```
AuthorizationOptions__Configuration__ServiceClientId=client
```
- the Client Secret value used for authorization in Keycloak (specify actual):
```
AuthorizationOptions__Configuration__ServiceClientSecret=secret
```
- the PostgreSQL DB connection string (this is the DB that stores the invitation block templates when creating an invitation in an Outlook email):
```
DatabaseOptions__Connection=Host=127.0.0.1;Port=5432;Database=exp
ress_meeting_db;Username=db_express;Password=Pass1234;Pooling=tru
e;Minimum Pool Size=50;Maximum Pool Size=100;Include Error
Detail=True;
```

Note. The parameters "Username=db_express" and "Password=Pass1234" must correspond to the credentials that were created during PostgreSQL configuration.

- the folder in the core-service container where logs will be saved (do not change):
```
Serilog__WriteTo__1__Args__configureLogger__WriteTo__0__Args__pat
h=/public/express-meeting-logs/coreservise.log
```
- the folder in the core-service container where logs received from clients will be saved (do not change):
```
Serilog__WriteTo__2__Args__configureLogger__WriteTo__0__Args__pat
h=/public/express-meeting-logs/client/plugin.log
```
- the service execution environment (do not change):
```
ASPNETCORE_ENVIRONMENT=Production
```

4. Prepare the template-service startup settings in the docker\template.env file.

   The file contains the following settings:

- The PostgreSQL DB connection string (this is the DB that stores the invitation block templates when creating an invitation in an Outlook email):
```
DatabaseOptions__Connection=Host=127.0.0.1;Port=5432;Database=exp
ress_meeting_db;Username=db_express;Password=Pass1234;Pooling=tru
e;Minimum Pool Size=50;Maximum Pool Size=100;Include Error
Detail=True;
```

Note. The parameters "Username=db_express" and "Password=Pass1234" must correspond to the credentials that were created during PostgreSQL configuration.

- A random set of characters used to generate the key that will sign the JWT access token for the service:
```
JWTOptions__Key=JFRINXV0LGIhLXlEVzlYdXlae316UU0zJzR9WVUtIWQ
```
- The value of the "Issuer" field when generating and checking the JWT token:
```
JWTOptions__Issuer=http://127.0.0.1
```
- The value of the "Audience" field when generating and checking the JWT token:
```
JWTOptions__Audience=http://127.0.0.1
```

- The folder in the template-service container where logs will be saved (do not change):

  ```
  Serilog__WriteTo__1__Args__path=/public/express-meeting-logs/templateservise.log
  ```
- The service execution environment (do not change):

  ```
  ASPNETCORE_ENVIRONMENT=Production
  ```

5. Run the docker containers from the docker folder:

   ```
   docker-compose --env-file ./.env -f ./docker-compose.yml up -d
   ```

   After starting the containers, if HTTPS addresses were specified in the ExpressOptions__Uri, AuthorizationOptions__Configuration__Url keys in the docker\core.env file, add the full certificate chain for the corresponding HTTPS connections to the express-meeting-core-service-main container if necessary (for example, if the Certificate Authority for the certificates used by the company is unavailable). An example of adding certificates to the container is described in Appendix 2.
6. Go to the Keycloak administrator web interface, navigate to the relevant realm and client used for user authentication in the Outlook add-in (this client was specified in the Client ID parameter).
7. In the Valid redirect URIs section, add the address(es) of the deployed core-service that will be specified in the client configuration files.
8. If necessary, add files from the following locations to the antivirus exclusions:

   ```
   /var/lib/docker
   ```

After starting the template-service container, when navigating to the address served by this container at the /front path (e.g.: http://localhost:7200/front), the login page for the add-in template administration system should open in the browser (Figure 5).

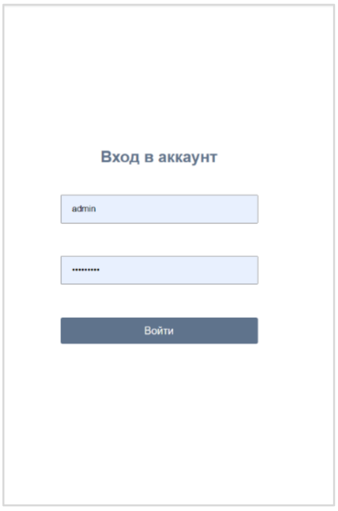


Figure 5

Note. The first login is performed using the admin account with a blank password.

STEP 4. DEPLOYING OUTLOOK ADD-IN ON CLIENT PCS

To deploy the Outlook add-in on client PCs:

1. Launch the ExpressMeeting_v_3.1.0.2.msi installer on the machine where Outlook is installed.

2. After installation, perform the following settings in the ExpressMeeting.dll.config[3] file for the add-in to work:
   - Address of the core-service (server part of the add-in) in the key (replace mycompany with the IP/DNS name of the Docker host machine where the core-service is deployed):

     ```
     <add key="ExpressMeetingUrl" value= "http://mycompany:7100/api/"
     />
     ```

     Note. The parameter can be initialized automatically during msi package installation. To do this, the ExpressMeetingUrl parameter must be passed in the command line.

     Example: ExpressMeeting_v_3.1.0.2.msi ExpressMeetingUrl="http://mycompany:7100/api/"

   - Timeout in seconds for establishing a connection when accessing Keycloak:

     ```
     <add key="Timeout" value="120"/>
     ```

     Note. The parameter can be initialized automatically during msi package installation. To do this, the Timeout parameter must be passed in the command line.

     Example: ExpressMeeting_v_3.1.0.2.msi Timeout="120"

   - If necessary, set the mode to prohibit conference type selection. The dropdown list for type selection (Public/Corporate/Trusted)[4] will be hidden in the meeting settings. Only Public meetings will always be created. To do this, add the following parameter to the configuration/appSettings section:

     ```
     <add key="DenyLinkType" value="true"/>
     ```

     Note. The parameter can be initialized automatically during msi package installation. To do this, the DenyLinkType parameter must be passed in the command line.

     Example: ExpressMeeting_v_3.1.0.2.msi DenyLinkType=true

   - If necessary, set the type of conference, created by default (possible values: Public, Trusts, Corporate). Meetings of the specified type will be created when the Create Conference button is clicked. To do this, add the following parameter to the configuration/appSettings section:

     ```
     <add key="DefaultLinkType" value="Corporate"/>
     ```

     Note. The parameter can be initialized automatically during msi package installation. To do this, the DefaultLinkType parameter must be passed in the command line.

     Example: ExpressMeeting_v_3.1.0.2.msi DefaultLinkType=Corporate

   - If necessary, in the ExpressMeeting.dll.config file, set the minimum level of password requirements (possible values: Public, Trusts, Corporate). The password switch will be unavailable for conferences of the specified type, as well as more public ones, when clicking the Create Conference button. To do this, add the following parameter to the configuration/appSettings section:

     ```
     <add key="LinkMinimumLevelRequirePassword" value="Corporate"/>
     ```

     Note. The parameter can be initialized automatically during msi package installation. To do this, the LinkMinimumLevelRequirePassword parameter must be passed in the command line.

     Example: ExpressMeeting_v_3.1.0.2.msi LinkMinimumLevelRequirePassword=Corporate

   - If necessary, enable background authentication token renewal so the user does not have to re-enter credentials after prolonged inactivity. To do this, add the following parameter to the configuration/appSettings section (Default value is false (background renewal is disabled)):

[3] The presumed path is c:\Program Files\Express\ExpressMeeting\ExpressMeeting.dll.config

[4] More details on the types of conferences created can be found in the Express CS user guides

```
<add key="BackgroundRefreshTokenProcess" value="true"/>
```

Note. The parameter can be initialized automatically during msi package installation. To do this, the BackgroundRefreshTokenProcess parameter must be passed in the command line.

Example: ExpressMeeting_v_3.1.0.2.msi BackgroundRefreshTokenProcess=true.

- If necessary, set the background authentication token renewal interval. The period is specified in "hours:minutes:seconds" format. For example, for 15 minutes, specify "00:15:00". To do this, add the following parameter to the configuration/appSettings section:

```
<add key="BackgroundRefreshTokenProcessPeriod" value="00:15:00"/>
```

Note. The parameter can be initialized automatically during msi package installation. To do this, the BackgroundRefreshTokenProcessPeriod parameter must be passed in the command line.

Example: ExpressMeeting_v_3.1.0.2.msi
BackgroundRefreshTokenProcessPeriod=00:15:00

- If it is necessary to enable Serilog internal logging (e.g., for diagnosing cases where main logs are not recorded), specify the full file path. To do this, add the following parameter to the configuration/appSettings section:

```
<add key="selfLogPath" value="C:\logs\serilogInternal.log"/>
```

3. In the serilogSettings.json[5] file, if telemetry is required, set the core-service address (server part of the add-in) in the key (replace mycompany with the IP/DNS name of the Docker host machine where the core-service is deployed):

```
Serilog.WriteTo[Name=Telemetry].Args.telemetryUrl="http://mycompan
y:7100/api/log/json"
```

Note. The parameter can be initialized automatically during msi package installation. To do this, the TelemetryUrl parameter must be passed in the command line. By default, the telemetry settings section is absent in serilogSettings.json.

Example: ExpressMeeting_v_3.1.0.2.msi

TelemetryUrl=" http://*mycompany*:7100/api/log/json "

4. The add-in supports different logging levels. To change the logging level in the serilogSettings.json[6] file, set the value of the Serilog.MinimumLevel.Default key to the required level. Possible logging level values are: Verbose, Debug, Information, Warning, Error, Fatal. Where Verbose means log everything, Debug means log Debug and above, Information means log Information and above, etc.

Note. The parameter can be initialized automatically during msi package installation. To do this, the LogLevel parameter must be passed in the command line.

Example: ExpressMeeting_v_3.1.0.2.msi LogLevel=Information

5. If necessary, add the following file locations to the antivirus exclusions on client PCs:
   - C:\Program Files\eXpress\ExpressMeeting (this path may vary for branded versions of the add-in);
   - C:\logs (if a different path was specified in Chapter 4, Step 4, item 2, then that path should be specified).

[5] The presumed path is c:\Program Files\Express\ExpressMeeting\serilogSettings.json.

[6] The presumed path is c:\Program Files\Express\ExpressMeeting\serilogSettings.json.

Chapter 5

ADMINISTRATOR WEB INTERFACE

This section describes the administrator web interface and how to work with it.

AUTHORIZING IN ADMINISTRATOR WEB INTERFACE

After installing the server part of the Outlook add-in, the login page for the add-in template administration system should open in the browser (Figure 6).

To authorize in the administrator console:

1. In the browser's address bar, enter the address of the administrator web interface (e.g., http://localhost:7200/front).

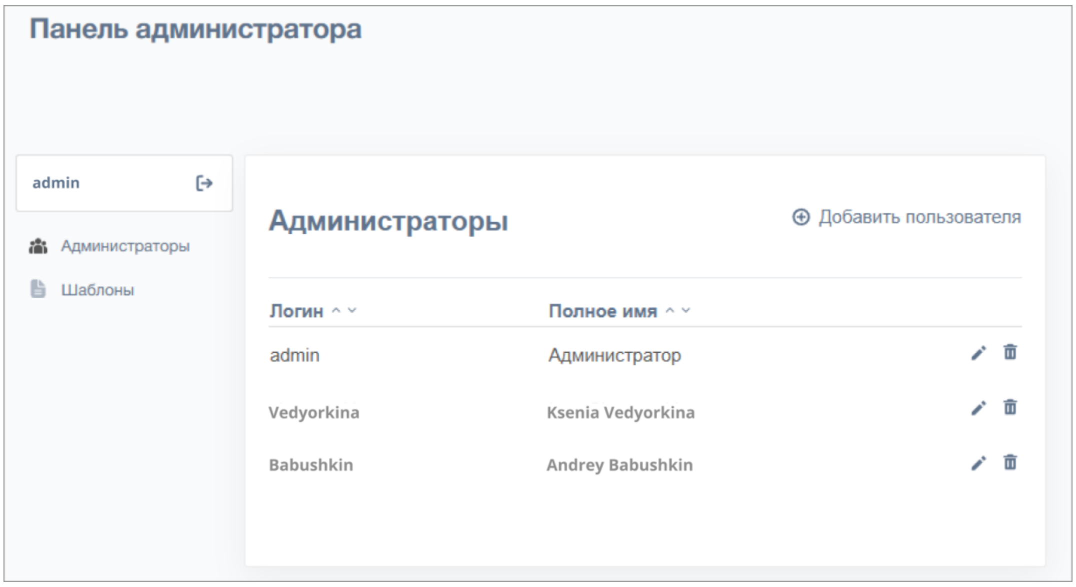
   The authorization window will open (Figure 7).
2. Enter the user account name and password into the corresponding fields.

   Note. The first login is performed using the admin account with an empty password.
3. Click the "Log in" button.

   Authorization will be granted.

DESCRIPTION OF THE ADMINISTRATOR PANEL INTERFACE

This section describes the administrator panel interface using the "Administrators" section interface as an example (Figure 6).



Логин	Полное имя
admin	Администратор
Vedyorkina	Ksenia Vedyorkina
Babushkin	Andrey Babushkin

Figure 6. "Administrators" Section Interface

MANAGING ADMINISTRATORS

The following operations are available to the administrator:

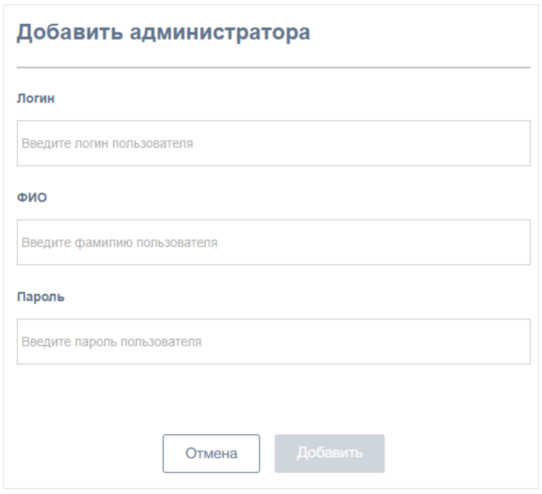
- adding an add-in administrator;
- editing add-in administrator data;
- deleting an add-in administrator.

To add an add-in administrator:

1. Click the "⊕ Add User" button.

   The administrator addition form will open (Figure 7).

2. Enter all necessary information in the form fields and click the “Add” button.


Добавить администратора

Логин

Введите логин пользователя

ФИО

Введите фамилию пользователя

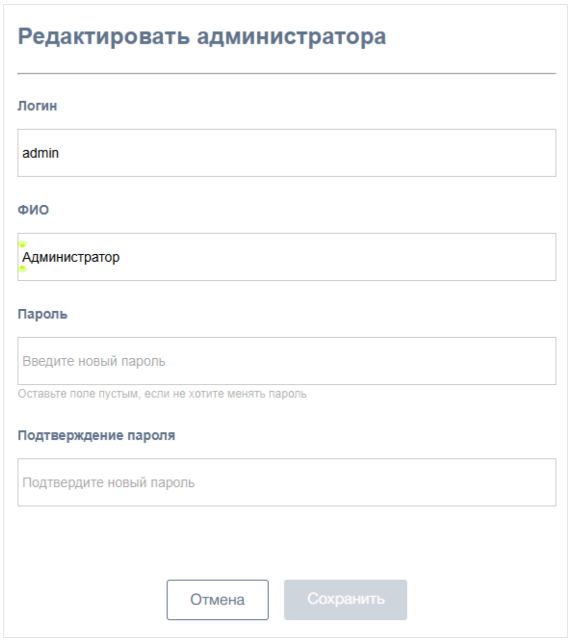
Пароль

Введите пароль пользователя

Figure 7. Adding an administrator

To edit administrator data:

1. Select an administrator from the list (Figure 6) and click the “Edit” button.
   The administrator data editing form will open (Figure 8):


Редактировать администратора

Логин

admin

ФИО

Администратор

Пароль

Введите новый пароль

Оставьте поле пустым, если не хотите менять пароль

Подтверждение пароля

Подтвердите новый пароль

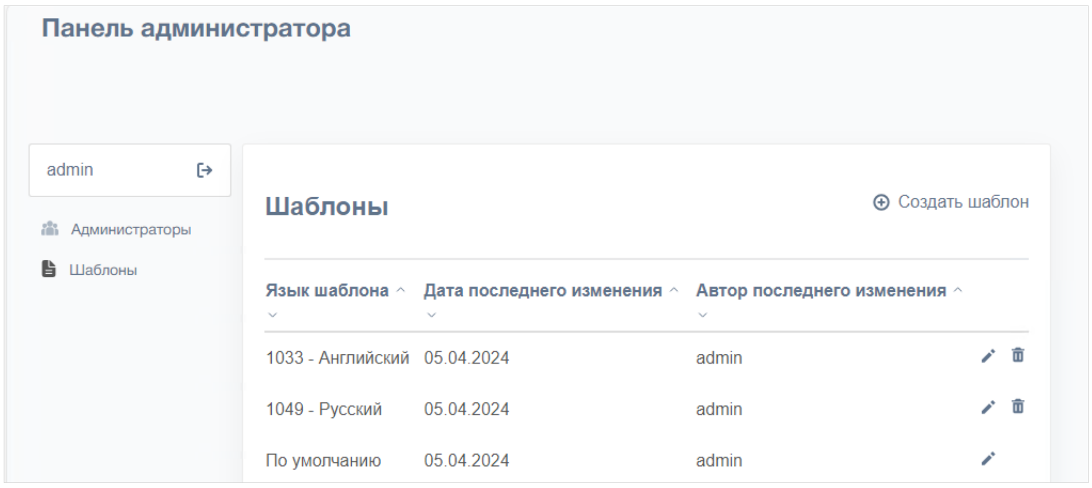
Figure 8. Making changes to administrator data

2. Make the necessary changes and click the “Save” button.

To delete an administrator, select the administrator from the list (Figure 6) and click the button.

TEMPLATES

In the “Templates” section (Figure 9), the administrator can create, edit (including in HTML mode), and delete invitation email templates.



Панель администратора

admin

Администраторы

Шаблоны

Шаблоны

Создать шаблон

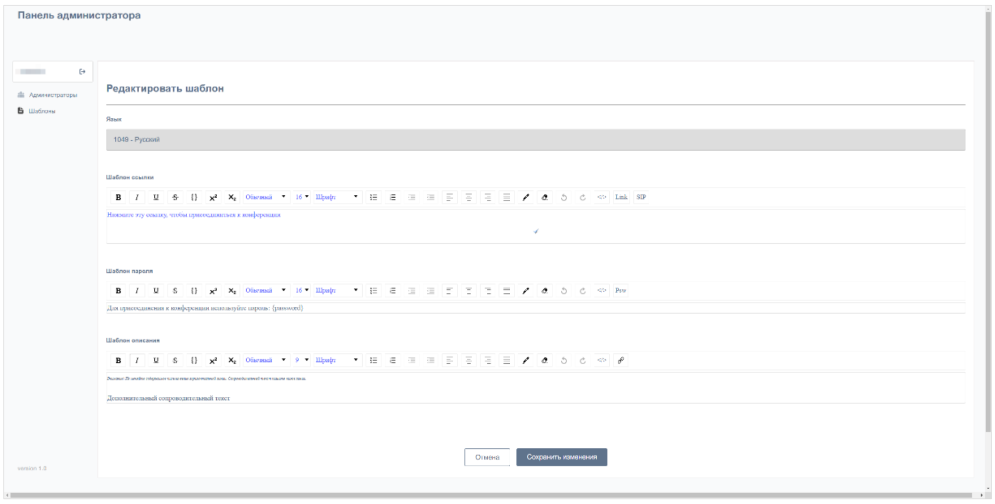
Язык шаблона	Дата последнего изменения	Автор последнего изменения
1033 - Английский	05.04.2024	admin
1049 - Русский	05.04.2024	admin
По умолчанию	05.04.2024	admin

Figure 9. The “Templates” section

To create a template:

1. Click the “Create Template” button

   A special visual template editor will open. It includes numerous functions: editing content, correcting font and text size, adding superscripts and subscripts, an HTML Editor, as well as buttons for inserting auto-substitution placeholders (conference links, current user's SIP numbers) (Figure 10).



Панель администратора

Редактировать шаблон

Язык: 1049 - Русский

Шаблон ссылки: Нажмите эту ссылку, чтобы присоединиться к конференции

Шаблон пароля: Для присоединения к конференции используйте пароль: {password}

Шаблон описания: Дополнительный сопроводительный текст

Figure 10. Visual Template Editor

2. Fill in the template fields and click the “Save Changes” button.

To edit a template:

1. Select a template from the list (Figure 9) and click the ✎ button.

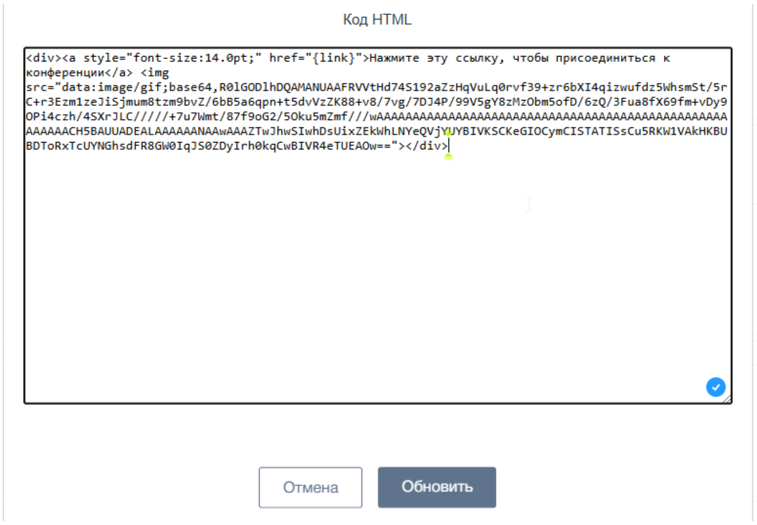
   The visual template editor will open (Figure 10).

2. Edit the template fields and click the “Save Changes” button.

To delete a template, select it from the list and click 🗑.

HTML EDITOR

To switch to the HTML Editor, click the "</>" in visual template editor menu. The editor window will look as follows (Figure 11):



Код HTML

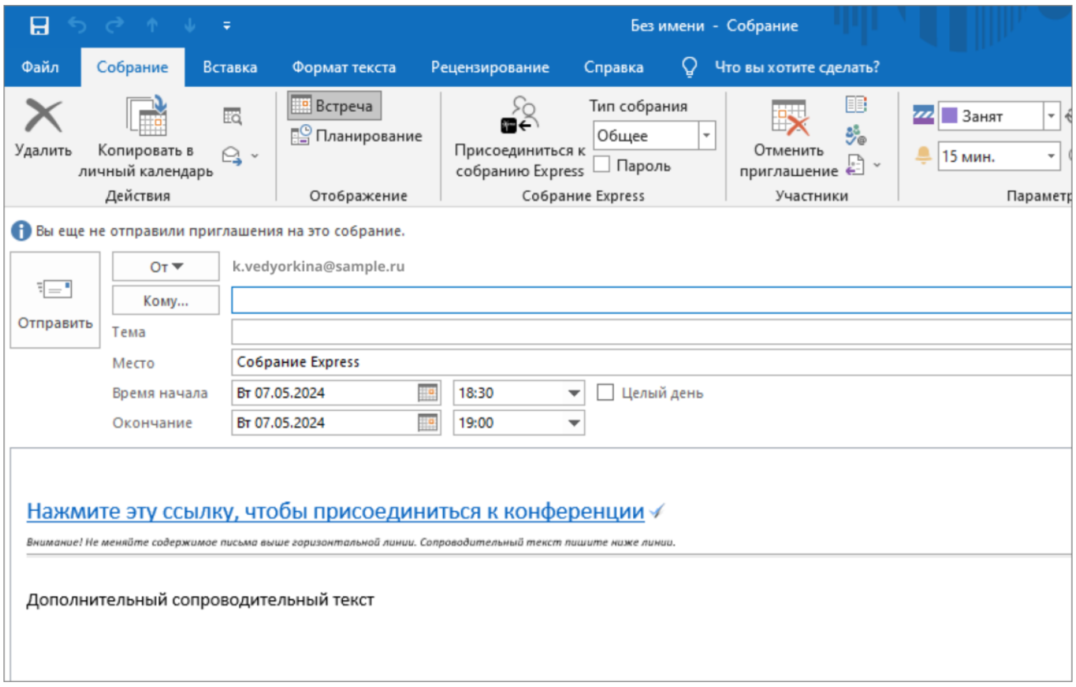
```
<div><a style="font-size:14.0pt;" href="{link}">Нажмите эту ссылку, чтобы присоединиться к
конференции</a> <img
src="data:image/gif;base64,R0lGODlhDQAMANUAAFRVVtHd74S192aZzHqVuLq0rvf39+zr6bXI4qizwufdz5WhsmSt/5r
C+r3Ezm1zeJiSjmum8tzm9bvZ/6bB5a6qpn+t5dvVzZK88+v8/7vg/7DJ4P/99V5gY8zMzObm5ofD/6zQ/3Fua8fX69fm+vDy9
OPi4czh/4SXrJLC/////+7u7Wmt/87f9oG2/5Oku5mZmf///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAACH5BAUUADEALAAAAAANAAwAAAZTwJhwSIwhDsUixZEkWhLNYeQVjYUYBIVKSCKeGIOCymCISTATISsCu5RKW1VAkHKBU
BDToRxTcUYNGhsdFR8GW0IqJS0ZDyIrh0kqCwBIVR4eTUEAOw=="></div>
```

Figure 11. HTML Editor

The editor fully supports HTML markup language. You can add images in it using the <img> tag and encode them in base64 format. For example:

```
<img src="data:image/gif;base64,R0lGODlhDQAMANUAAFRVVtHd74S192aZzHqVuLq0
rvf39+zr6bXI4qizwufdz5WhsmSt/5rC+r3Ezm1zeJiSjmum8tzm9bvZ/6bB5a6qpn+t5dvV
zZK88+v8/7vg/7DJ4P/99V5gY8zMzObm5ofD/6zQ/3Fua8fX69fm+vDy9OPi4czh/4SXrJLC
/////+7u7Wmt/87f9oG2/5Oku5mZmf///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAACH5BAUUADEALAAAAAANAAwAAAZTwJhwSIwhDsUixZEkWhLNYeQVjYUY
BIVKSCKeGIOCymCISTATISsCu5RKW1VAkHKBUBDToRxTcUYNGhsdFR8GW0IqJS0ZDyIrh0kq
CwBIVR4eTUEAOw==">
```

In the invitation email, this will be displayed as follows (Figure 12):



Вы еще не отправили приглашения на это собрание.

От: k.vedyorkina@sample.ru

Место: Собрание Express

Время начала: Вт 07.05.2024 18:30

Окончание: Вт 07.05.2024 19:00

Нажмите эту ссылку, чтобы присоединиться к конференции

Внимание! Не меняйте содержимое письма выше горизонтальной линии. Сопроводительный текст пишите ниже линии.

Дополнительный сопроводительный текст

Figure 12. Result of working in the HTML Editor

There are currently two templates—in Russian and English. The template is selected by the client part of the add-in and depends on the MS Outlook language locale. The Russian template will be automatically applied in Russian MS Outlook, and the English template in English MS Outlook. Manual template selection is not provided.

Chapter 6

DIAGNOSING FAULTS IN THE OUTLOOK ADD-IN

This section contains recommendations for performing correct diagnostics.

Attention! This section does not contain instructions for fixing identified malfunctions.

The Operating Principle of the Outlook Add-in

1. When the button "Create meeting" is pressed in Outlook, a request is generated and sent to the Keycloak server.
2. Authentication and authorization are performed on the Keycloak server.
3. If authentication and authorization are successful, Keycloak transfers the Token to express-meeting-core-service.
4. The express-meeting-core-service sends the email template from the DB to the Outlook add-in and sends a request to the chat bot to create a template for the future conference.
5. The Conference Notifier Bot creates a template for the future conference in Express CS.
6. The client-side Outlook add-in creates an email in Outlook from the template, inserting a link to the future conference.
7. The user specifies the participants, date, time, and name of the future conference by filling in the corresponding fields in the Outlook email.
8. When the user clicks the "Send" button, the Outlook add-in generates and sends a request to express-meeting-core-service to modify the previously created conference, taking into account the parameters filled in by the user.
9. If authentication and authorization are successful, express-meeting-core-service sends a request to the chat bot to modify the future conference template, taking into account the parameters filled in by the user.
10. The chat bot changes the conference parameters in Express CS to the current ones.

If the operating principle of the add-in described above is not fulfilled, use the troubleshooting steps described below.

STEP 1. CHECKING CONFERENCE NOTIFIER BOT FUNCTIONALITY

First, check the functionality of the Conference Notifier Bot on the eXpress server side, because if the chatbot does not accept or create conferences in eXpress, further configuration and diagnostic actions for other system components will be useless.

To check the functionality of the Conference Notifier Bot:

1. Connect via SSH to the eXpress Corporate Server (CTS).
2. Execute a request to the Conference Notifier Bot using the Curl program.
3. Execute a request to the Conference Notifier Bot via the mobile application chat or the web/desktop version of the eXpress application:
   - If the request to the Conference Notifier Bot via the Curl program and the chat in eXpress is successful, proceed to the next step.
   - If the request to the Conference Notifier Bot via the Curl program or the chat in eXpress was unsuccessful, you need to restore the functionality (check the settings) of the Conference Notifier Bot and proceed to the next step.

   Steps for checking the functionality of the Conference Notifier Bot using the Curl program are described in Chapter 4 "Installing the Outlook Add-in" of this instruction:

- If chatbot security was **not** configured, see items 5-7 of the section "Step 1. Enabling Conference Notifier Bot API Without Password";
- If chatbot security was configured, see items 7-9 of the section "Step 2. Securing Conference Notifier Bot".

STEP 2. CHECKING AVAILABILITY OF CONFERENCE NOTIFIER BOT FROM DOCKER SERVER

Next, it is recommended to check availability of Conference Notifier Bot from Docker server.

To check availability of the Conference Notifier Bot from the Docker server:

1. Access the Docker server where the express-core-service application is installed, using SSH or another method.
2. Perform the diagnostics similarly to Step 1 "Checking Conference Notifier Bot Functionality" — above.

STEP 3. CHECK EXPRESS-MEETING-CORE-SERVICE APPLICATION AVAILABILITY FROM CLIENT WORKSTATION

Next, it is recommended to check availability of the express-core-service application on Docker server from the client workstation.

To check availability of Docker server from the client workstation:

1. Connect via RDP (or another method) to the client workstation where the Microsoft Outlook application and the Outlook add-in are installed.

   Connect via RDP (or another method) to the client workstation where the Microsoft Outlook application and the Outlook add-in are installed. In PowerShell ISE, run the diagnostic script from Appendix 1 and edit the variable values: the Outlook user account, the link to Docker, and the token received from Keycloak. The Keycloak token can be obtained by requesting it from the Customer's Keycloak Administrator.
2. If the script executed with errors, analyze and fix their causes. Probable causes of the problem in the Docker and express-core-service settings are: lack of network access, or authentication and authorization issues. Rerun the script until it executes without errors.
3. If the script executed without errors and returned a conference ID, which can be found in the administration console of the Express CS CTS-server, then the Conference Notifier Bot is considered available from the Docker server.

   Proceed to the next step.

STEP 4. GENERAL DIAGNOSTICS OF OUTLOOK ADD-IN

If previous 3 steps have been completed successfully, but Outlook add-in still does not work, check correctness of settings in client software configuration file (see section "Step 3. Deploying express-core-service and template-core-service in Docker" of this instruction).

If client software configuration file is correct, examine log files from client applications, described in section "Step 3. Deploying express-core-service and template-core-service in Docker" of this instruction, and server log files, described in section "Step 2. Preparing the Database". Eliminate causes of errors.

If all steps listed above have not helped, contact manufacturer's technical support for assistance.

Appendix 1

DIAGNOSTIC SCRIPT NO. 1

This document does not contain instructions for fixing identified malfunctions but explains the diagnostic script's operational principle.

```
$headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/json")
$headers.Add("Authorization", token)

# fill in the creator below
$body = @"
{`"Creator`":`"Express_User@Your_domain.com`",`"LanguageId`":1049,`"isRe
current`":false,`"`$type`":`"MeetingCreateRequest`"}
"@


#  enter the Docker express-core-service address below; the same link
must be present in the client-side add-in settings.

$response = Invoke-RestMethod 'http://
Your_Docker:Your_port/api/meetings' -Method 'POST' -Headers $headers -
Body $body
$response | ConvertTo-Json
```

Appendix 2

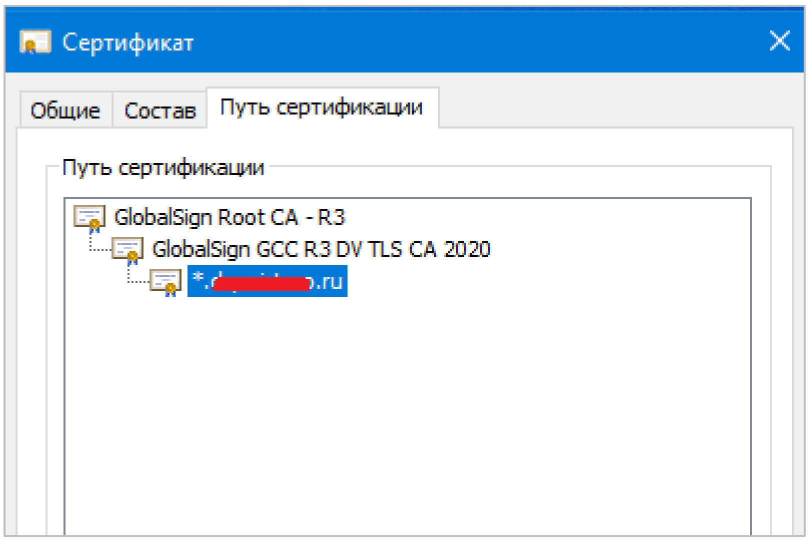
ADDING A CERTIFICATE TO THE EXPRESS-MEETING-CORE-SERVICE-MAIN CONTAINER

This section describes a typical solution for installing certificates into a Docker container. The procedure may vary depending on the infrastructure and architectural solution.

Important! For correct SSL operation, all certificates in the certificate chain (root certificate, intermediate certificates, end certificate) should be placed in the express-meeting-core-service-main container.

To add the certificate to the container:

1. Gather all certificates in the chain as CRT files (Figure 13):



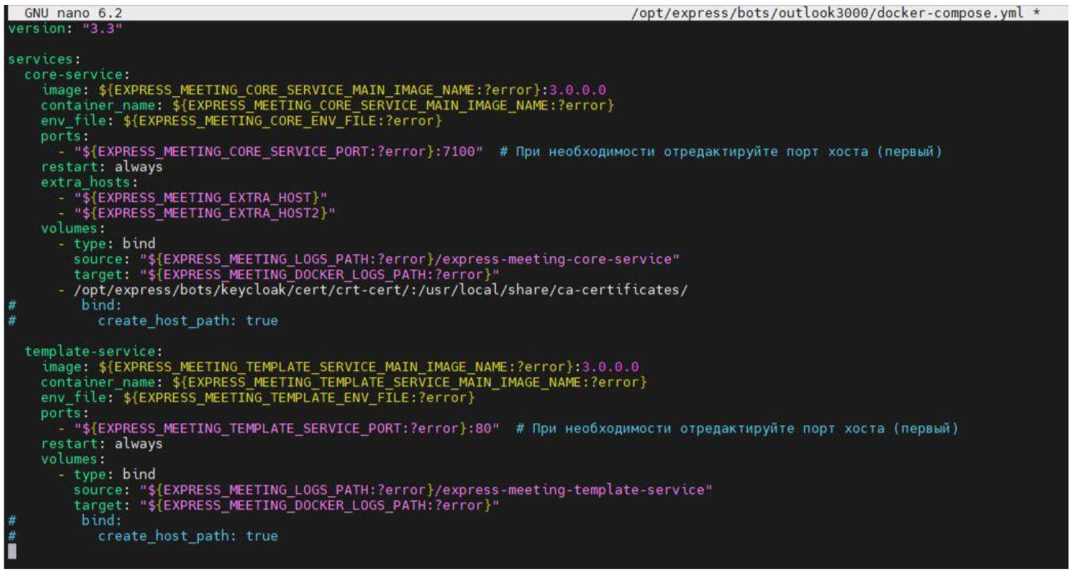
GlobalSign Root CA - R3
GlobalSign GCC R3 DV TLS CA 2020
*.[redacted].ru

Figure 13

2. Create a folder on the Docker server where the express-meeting-core-service-main container is deployed. Example folder name: /opt/express/bots/cert/.
3. Move the certificate files gathered in step 1 to the folder created in step 2.
4. Grant the appropriate access rights to the /opt/express/bots/cert/ folder.
5. In the docker-compose.yml file, add the following parameter to the volumes section in the parameters of the express-meeting-core-service-main container (Figure 14):

```
- /opt/express/bots/cert/:/usr/local/share/ca-certificates/
```

Important! The first part of the string, /opt/express/bots/cert/, can be anything and must correspond to the path from step 2. The second part of this string, :/usr/local/share/ca-certificates/, is constant and must not be changed. The express-meeting-core-service-main takes all necessary certificates from the specified folder.



```
  GNU nano 6.2                                          /opt/express/bots/outlook3000/docker-compose.yml *
version: "3.3"

services:
  core-service:
    image: ${EXPRESS_MEETING_CORE_SERVICE_MAIN_IMAGE_NAME:?error}:3.0.0.0
    container_name: ${EXPRESS_MEETING_CORE_SERVICE_MAIN_IMAGE_NAME:?error}
    env_file: ${EXPRESS_MEETING_CORE_ENV_FILE:?error}
    ports:
      - "${EXPRESS_MEETING_CORE_SERVICE_PORT:?error}:7100"  # При необходимости отредактируйте порт хоста (первый)
    restart: always
    extra_hosts:
      - "${EXPRESS_MEETING_EXTRA_HOST}"
      - "${EXPRESS_MEETING_EXTRA_HOST2}"
    volumes:
      - type: bind
        source: "${EXPRESS_MEETING_LOGS_PATH:?error}/express-meeting-core-service"
        target: "${EXPRESS_MEETING_DOCKER_LOGS_PATH:?error}"
      - /opt/express/bots/keycloak/cert/crt-cert/:/usr/local/share/ca-certificates/
#        bind:
#          create_host_path: true

  template-service:
    image: ${EXPRESS_MEETING_TEMPLATE_SERVICE_MAIN_IMAGE_NAME:?error}:3.0.0.0
    container_name: ${EXPRESS_MEETING_TEMPLATE_SERVICE_MAIN_IMAGE_NAME:?error}
    env_file: ${EXPRESS_MEETING_TEMPLATE_ENV_FILE:?error}
    ports:
      - "${EXPRESS_MEETING_TEMPLATE_SERVICE_PORT:?error}:80"  # При необходимости отредактируйте порт хоста (первый)
    restart: always
    volumes:
      - type: bind
        source: "${EXPRESS_MEETING_LOGS_PATH:?error}/express-meeting-template-service"
        target: "${EXPRESS_MEETING_DOCKER_LOGS_PATH:?error}"
#        bind:
#          create_host_path: true
```

Figure 14

6. Stop the Outlook add-in containers by running the following command from the folder with the configuration files:

```
docker-compose down
```

7. Start the docker containers from the folder:

```
docker-compose --env-file ./.env -f ./docker-compose.yml up -d
```

8. Access the express-meeting-core-service-main container using the command:

```
docker exec -it express-meeting-core-service-main bash
```

9. Update the list of certificates with the command:

```
update-ca-certificates
```

10. Exit the express-meeting-core-service-main container using the command:

```
exit
```

11. Try creating a conference using the client software.
12. Check the system for the absence of SSL-related errors using the command:

```
docker logs --tail=200 -f  express-meeting-core-service-main
```

Appendix 3

NETWORK INTERACTION

Table 4. Single CTS or Split CTS

No.	Source	Destination	Port and Protocol	Interaction Description
1.	Internal User (Client PC)	express-core-service (Docker Host)	7100\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with express-core-service001 on the Docker Host server (standard network port specified)
2.	express-core-service (Docker Host)	CTS Front (CTS Single)	443\TCP	Interaction of express-core-service on Docker Host with Conference Notifier Bot API on CTS Front (CTS Single) server
3.	Template Administrator (Client PC)	express-template-service (Docker Host)	7200\TCP	Interaction of the Template Administrator with express-template-service001 on Docker Host via the web interface (standard network port specified)
4.	express-core-service (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-core-service001 on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
5.	express-template-service (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-template-service on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
6.	Internal User (Client PC)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with Keycloak on the Keycloak server (standard network port specified)
7.	express-core-service (Docker Host)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of express-core-service001 on Docker Host with Keycloak on the Keycloak server (standard network port specified)

Note. The table specifies standard network ports for interaction between client PCs, express-core-service, and Keycloak (SSL is configured separately and is not considered in this table).

Table 5. Multiple CTS without ETS

No.	Source	Destination	Port and Protocol	Interaction Description
1.	Internal User on CTS01 (Client PC)	express-core-service001 (Docker Host)	7100\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with express-core-service001 on Docker Host (standard network port specified)
2.	Internal User on CTS02 (Client PC)	express-core-service002 (Docker Host)	7300\TCP	Interaction of the Outlook add-in on the user's PC on CTS02 with express-core-service002 on Docker Host (standard network port specified)
3.	Template Administrator (Client PC)	express-template-service (Docker Host)	7200\TCP	Interaction of the Template Administrator with express-template-service on Docker Host via the web interface (standard network port specified)
4.	express-core-service001 (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-core-service001 on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
5.	express-template-service (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-template-service on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
6.	express-core-service002 (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-core-service002 on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
7.	express-core-service001 (Docker Host)	CTS Front (CTS Single)01	443\TCP	Interaction of express-core-service001 on Docker Host with Conference Notifier Bot API on CTS Front (CTS Single)01 server
8.	express-core-service002 (Docker Host)	CTS Front (CTS Single)02	443\TCP	Interaction of express-core-service002 on Docker Host with Conference Notifier Bot API on CTS02 Front (CTS02 Single)02 server
9.	express-core-service001 (Docker Host)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of express-core-service001 on Docker Host with Keycloak on the Keycloak server (standard network port specified)
10.	Internal User (Client PC)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with Keycloak on the Keycloak server (standard network port specified)
11.	express-core-service002 (Docker Host)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of express-core-service002 on Docker Host with Keycloak on the Keycloak server (standard network port specified)
12.	Internal User (Client PC)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of the Outlook add-in on the user's PC on CTS02 with Keycloak on the Keycloak server (standard network port specified)

Note. The table specifies standard network ports for interaction between client PCs, express-core-service, and Keycloak (SSL is configured separately and is not considered in this table).

Table 6. Multiple CTS with ETS

No.	Source	Destination	Port and Protocol	Interaction Description
1.	Internal User on CTS1 (Client PC)	express-core-service (Docker Host)	7100\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with express-core-service001 on Docker Host (standard network port specified)
2.	Internal User on CTS2 (Client PC)	express-core-service (Docker Host)	7100\TCP	Interaction of the Outlook add-in on the user's PC on CTS02 with express-core-service001 on Docker Host (standard network port specified)
3.	Template Administrator (Client PC)	express-template-service (Docker Host)	7200\TCP	Interaction of the Template Administrator with express-template-service on Docker Host via the web interface (standard network port specified)
4.	express-core-service001 (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-core-service001 on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
5.	express-template-service (Docker Host)	PostgreSQL DB (DB server)	5432\TCP	Interaction of express-template-service on Docker Host with PostgreSQL DB on the DB server (standard network port specified)
6.	express-core-service (Docker Host)	ETS EXPRESS Server	443\TCP	Interaction of express-core-service on Docker Host with Conference Notifier Bot API on the ETS EXPRESS Server
7.	express-core-service (Docker Host)	ETS EXPRESS Server	443\TCP	Interaction of express-core-service001 on Docker Host with Conference Notifier Bot API on the ETS EXPRESS Server
8.	express-core-service001 (Docker Host)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of express-core-service001 on Docker Host with Keycloak on the Keycloak server (standard network port specified)
9.	Internal User (Client PC)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with Keycloak on the Keycloak server (standard network port specified)
10.	Internal User (Client PC)	Keycloak Functionality (Docker Host)	8080\TCP	Interaction of the Outlook add-in on the user's PC on CTS02 with Keycloak on the Keycloak server (standard network port specified)

Note:

- The table specifies standard network ports for interaction between client PCs, express-core-service, and Keycloak (SSL is configured separately and is not considered in this table).
- Network interaction between EXPRESS ETS servers and EXPRESS CTS servers is described in detail in “Express. Communication System. Administrator's Guide. Volume 1. Installation” (https://express.ms/admin_guide_install.pdf).

Appendix 4

COMMON ISSUES

Description: If problems occur with PostgreSQL DB connection configuration, the following message appears in the logs:

```
"An error occurred while processing your
request.","status":500,"detail":"System.InvalidOperationException: An
exception has been raised that is likely due to a transient failure.\n -
--> Npgsql.NpgsqlException (0x80004005)
```

Possible Solution:

Edit the environment parameters in the appsettings.Production.json[7] file:

- Set Minimum Pool Size=0.
- If the previous step does not help solve the problem, set Pooling=False.

If the error persists, contact the Express CS technical support.

[7] Suggested path: IIS\template-service\appsettings.Production.json.