UNLIMITED PRODUCTION

eXpress Communication System

Microsoft Outlook Add-in

Installation Guide

Active Directory Version 3.1.0.2

© Unlimited Production Company, 2025. All rights reserved.

All copyrights to the operational documentation are reserved. This document, or any part thereof, in printed or electronic form, may not be copied or transferred to third parties for commercial purposes without the specific written permission of the Unlimited Production Company.

The information contained in this document may be changed by the developer without special notice, which does not constitute a violation of obligations to the user on the part of the Unlimited Production Company.

The server addresses, configuration file values, and user account data specified in the document are provided as examples and are for informational purposes only. The user data, including biometric data, are fictitious and do not contain personal data.

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 family of operating systems
API	Application Programming Interface — an interface for the interaction of programs and applications
Chatbot	The Conference Notifier Bot Chatbot
CTS	Corporate Transport Server
Conference Creator	User who creates and can edit an Express conference using the Outlook add-in
DB	Database
DBMS	Database Management System
ETS	Enterprise Transport Server
Express CS, Express, System	The “Express” Communications System
express-core-service	Server application for the Outlook add-in, providing core functionality (creating and modifying conferences)
express-template-service	Server application for the Outlook add-in, providing additional functionality (creating and modifying email templates, text, logo, signature, etc.)
Invited Participant	User(s) invited by the Express conference creator using the Outlook add-in
JSON	Text-based data exchange format based on JavaScript
Log	An entry in the server and/or client event journal
MS Exchange	Microsoft mail server
MS Outlook	Microsoft mail client, part of the MS Office application suite
OS	Operating System
PC	Personal Computer
Single CTS	Single Corporate Transport Server
Split CTS	Split Corporate Transport Server: Front CTS and Back CTS

Chapter 1

GENERAL INFORMATION

This guide describes the installation process for the Express CS add-in (hereinafter – the add-in) for Microsoft Outlook (hereinafter – Outlook), the actions of the add-in's server-side administrator, and add-in operation diagnostics.

Important! Performing the operations described in this guide requires the following competencies:

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

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

- Outlook client version 2013 or later;
- IIS version 10.0 or later (during IIS installation, select web-server → security → windows authentication);
- the IIS server must have the following package installed: https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-aspnetcore-7.0.11-windows-hosting-bundle-installer;
- for seamless authentication, the IIS server must be domain-joined, and appropriate access rights must be configured.
- database requirements:
  - PostgreSQL version 13.11 or later (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.
- Outlook must be used under a domain account with a mailbox matching the email address specified in the AD account;
- an installed and configured Express CS solution, version 3.8 or later;
- the Conference Notifier Bot chat bot must be configured in the Express CS; Configuration requirements and methods for verifying correct bot setup are described in Chapter 3, "Installing and Configuring the Conference Notifier Chat Bot".

The PostgreSQL DBMS can reside on a dedicated server running Windows OS or Linux OS. If a pre-installed PostgreSQL DBMS is unavailable and the deployed solution architecture includes an Express CS Bot server, you can use the PostgreSQL Docker container included with the Bot server to create the database. The PostgreSQL database can also be created on the IIS server where the Outlook add-in will be deployed.

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 Linux OS must meet the following technical requirements (Table 1):

Table 1

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

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 cores
RAM	2 GB
Operating System	Windows Server 2016 or later
Hard Disk	At least 40 GB

If the virtual machine will not host other services, the IIS server for the Outlook add-in must meet the following technical requirements (Table 3):

Table 3

Component	Parameters
Processor	64-bit 1.4 GHz processor 2 cores
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 (**Ошибка! Источник ссылки не найден.**).

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.

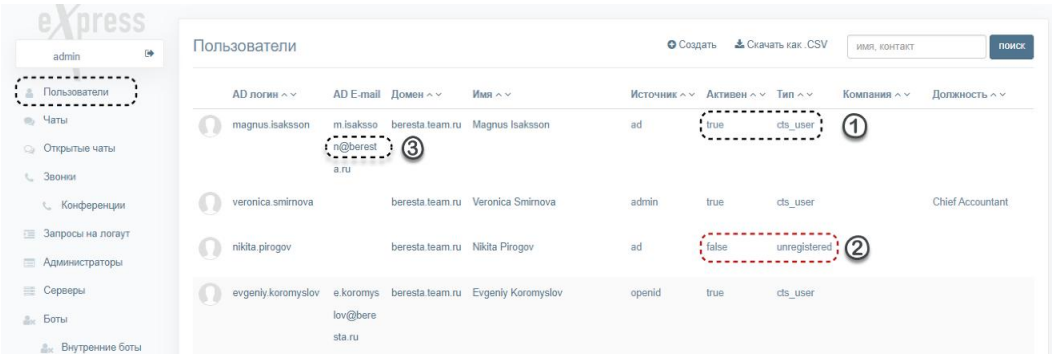

Figure 1. User Status: 1 – Active; 2 – Inactive

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 (**Ошибка! Источник ссылки не найден.**, 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 AND NETWORK INTERACTION

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.

When using more than one instance of express-core-service or express-template-service on a single IIS server, it is better to configure different log write paths for each server application (each server application should have its own log folder).

SINGLE CTS OR SPLIT CTS

If the Express CS contains a single CTS server, the server-side of the add-in should be deployed as a single instance (Figure 2). The network interaction description is provided in Table 4 Appendix 3.

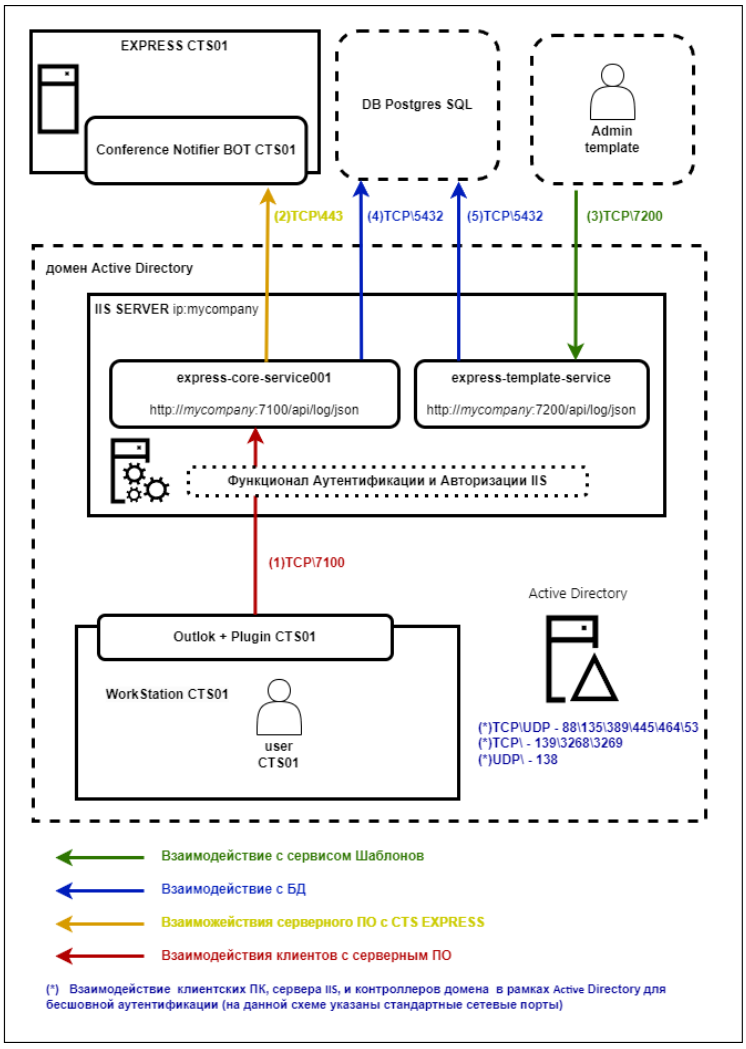

Figure 2. Typical scheme: 1 CTS, 1 express-core-service, 1 express-template-service

MULTIPLE CTS

If the Express CS contains multiple CTS servers without an ETS server, the server-side of the add-in should be deployed separately for each CTS server (this can be done on a single IIS with different ports or on multiple IIS servers) (Figure 3). The network interaction description is provided in Table 5 Appendix 3.

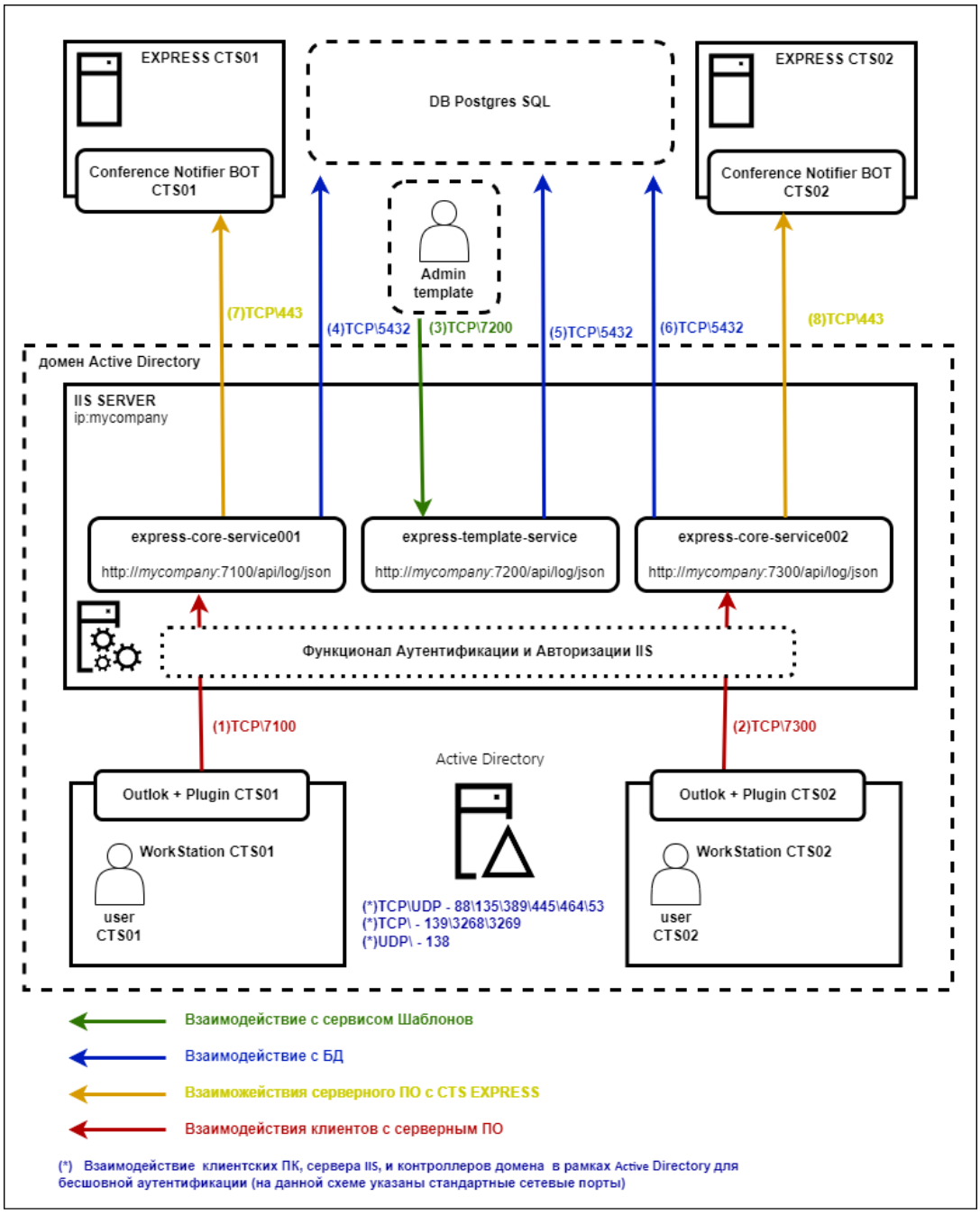

Figure 3. Typical scheme: 2 CTS, 2 express-core-service, 1 express-template-service

MULTIPLE CTS WITH ETS

If the Express CS contains multiple CTS servers united by an ETS server (Figure 4), the server-side of the add-in should be deployed only for the ETS server. The network interaction description is provided in Table 6 Appendix 3.

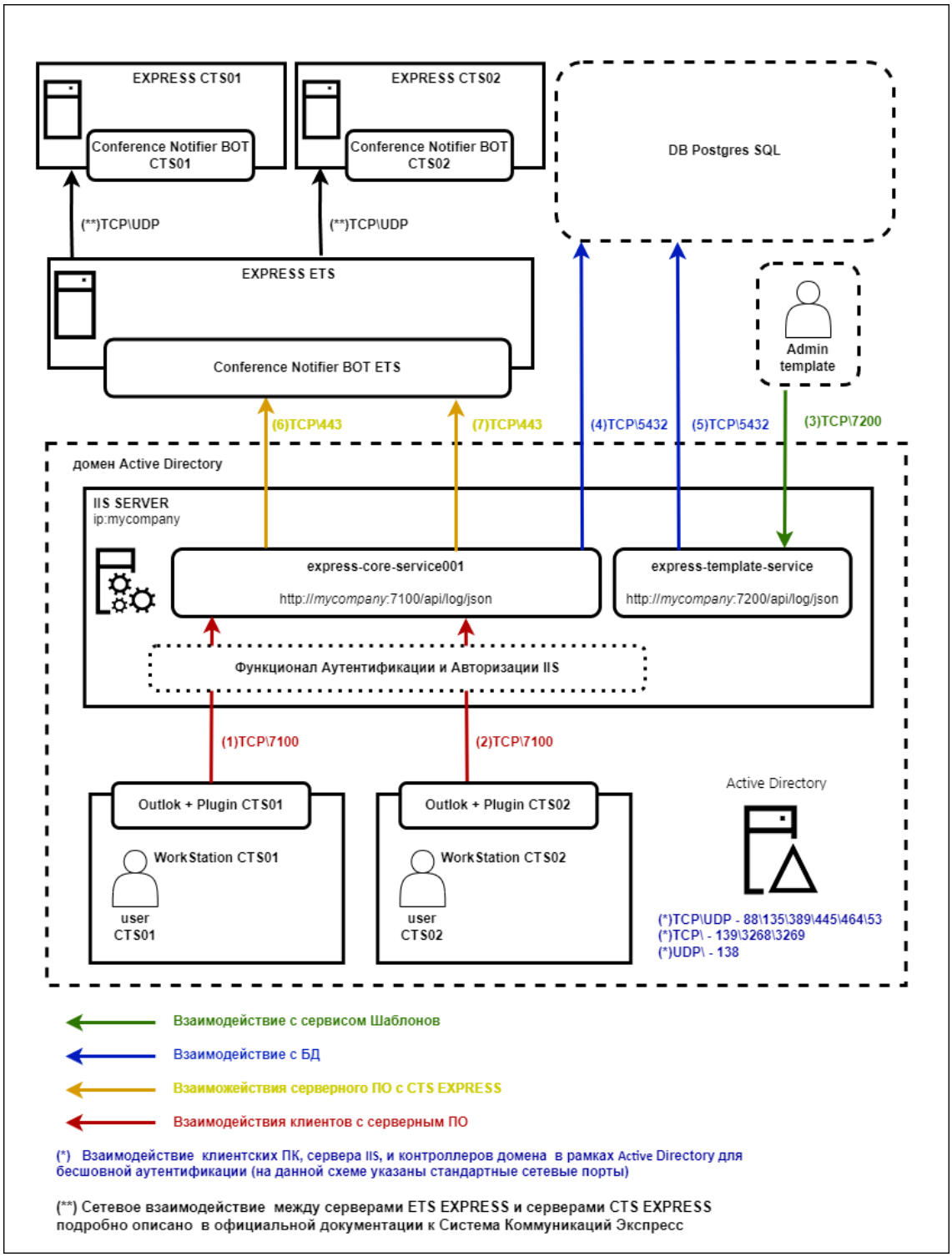

Figure 4. Typical scheme: 1 ETS, 2 CTS, 1 express-core-service, 1 express-template-service

Note. The network interaction between the EXPRESS ETS servers and the EXPRESS CTS servers (Figure 4) is described in detail in "Express. Communication System. Administrator Guide. Volume 1. Installation" at https://express.ms/admin_guide_install.pdf

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 perimeter 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 the following command 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 the "Search" button.

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

[1] Предполагаемый путь /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, containing the previously created BEARER TOKEN, 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 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=%2F%2FAD (select the current product version).

STEP 1. DEPLOYING EXPRESS-CORE-SERVICE IN IIS

If the service was already deployed as part of a previous add-in version, follow these steps:

- Ensure that the account used for the update has all necessary file system permissions.
- Replace the service files with the files from the distribution's IIS\core-service folder (except for the configuration file appsettings.Test.json. Do not replace this file! Otherwise, you will have to reconfigure the settings described in item 7).
- Rename the saved appsettings.Test.json file to appsettings.Production.json (optional).
- On the IIS server, change the value of the ASPNETCORE_ENVIRONMENT parameter to Production (setting this value is described in item 6) (optional).
- Perform item 9 to update the database structure.

To deploy express-core-service in IIS:

1. Create an AppPool in IIS (Figure 5):

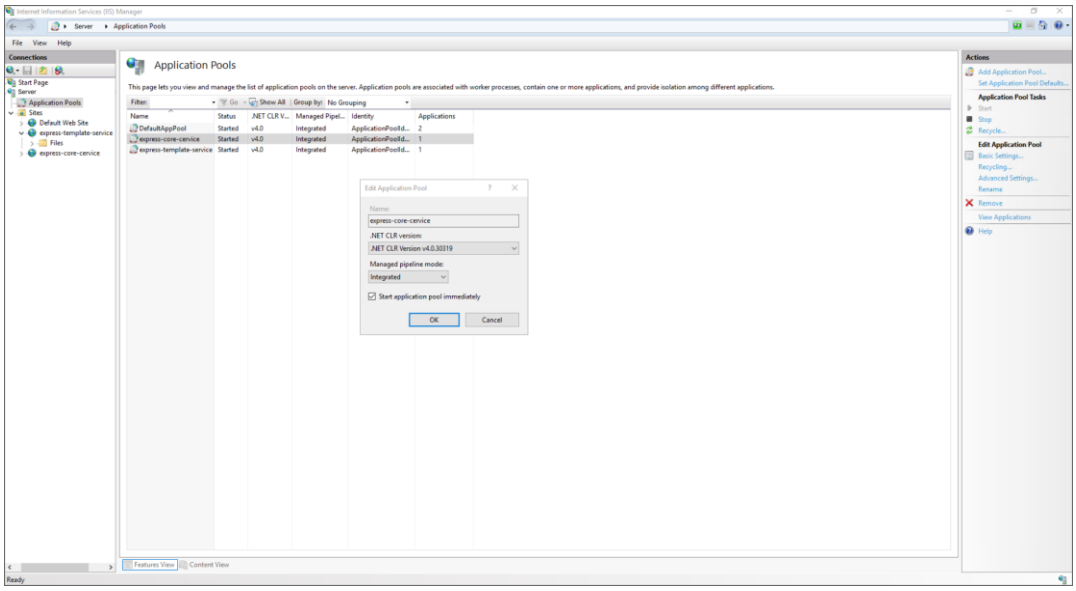

Figure 5

2. Create the express-core-service website.

   When creating, the path to the IIS\core-service folder must be specified in the PhysicalPath field.

3. In the Application pool field, select the pool created earlier.
4. In the Port field, specify the port for client communication with the IIS server.

5. Navigate to “Home Node” → “Feature Delegation” → “Custom Site Delegation”. In the window that opens for the express-core-service website, grant read/write permissions for the Authentication – Windows and Authentication – Anonymous parameters (Figure 6).

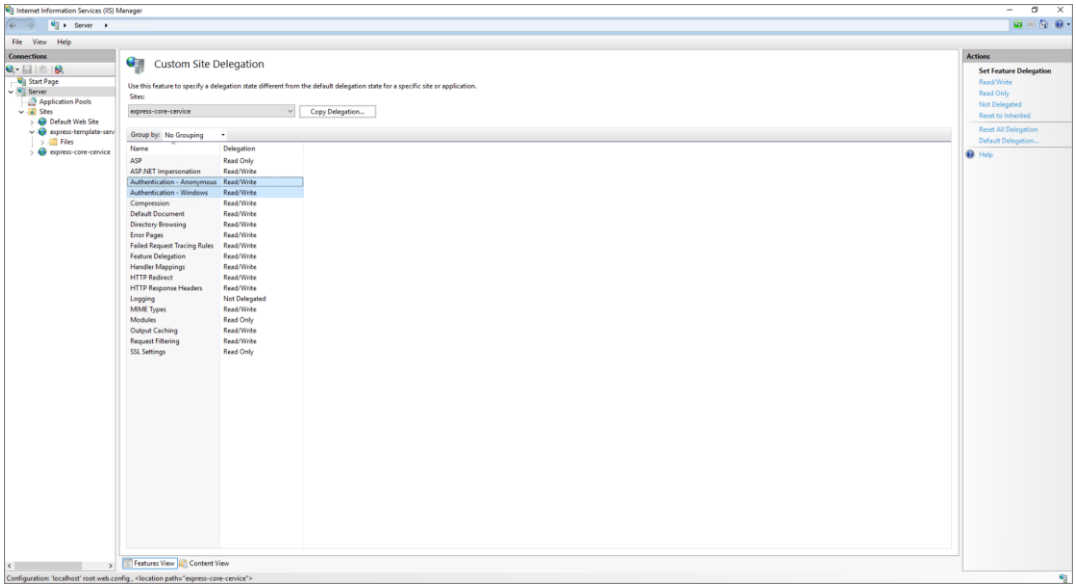

Figure 6

6. For the express-core-service site node, use the Configuration Editor to set the runtime environment: system.webServer/aspNetCore → environmentVariables From ApplicationHost.config (Figure 7). Set the corresponding parameters:

```
name: ASPNETCORE_ENVIRONMENT
value: Production
```

Then close the environmentVariables window and click Apply in the Configuration Editor:

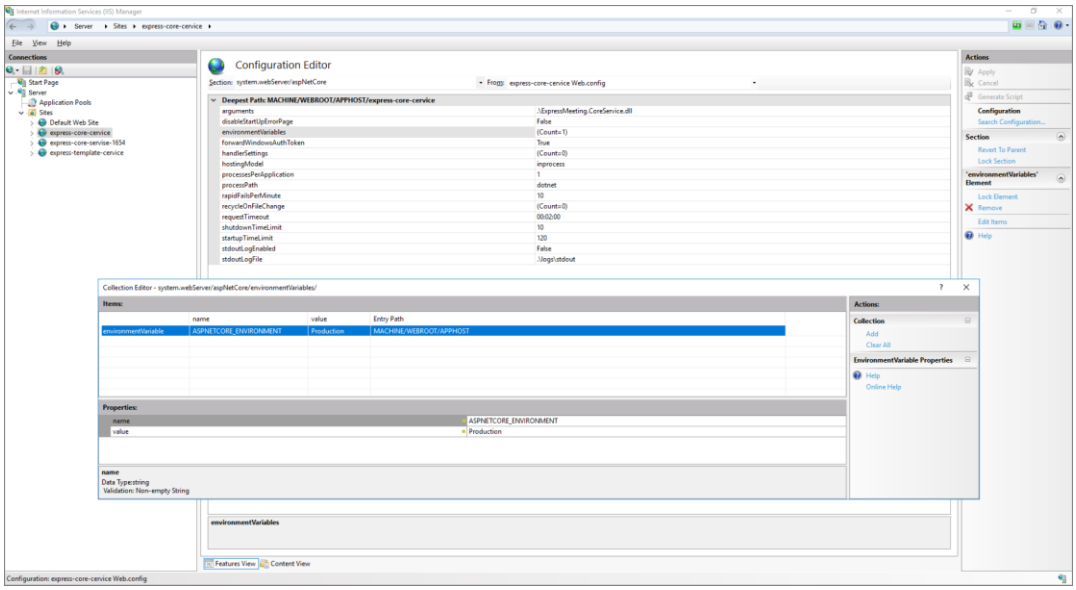

Figure 7

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

```
"ExpressOptions": {
 "AuthenticationScheme": "StaticToken",
 "Token": "Bearer dfgdf509usds0f9n0ud",
 "Uri": "https://.../api/v1/conference_bot/conferences/",
 "Timeout": 120
  },
```

```
"DatabaseOptions": {
 "Connection": "...;"
  },
"ADOptions": {
 "DirectorySearcherDomain": ["mydom1.local","mydom2.local"],
 "DirectorySearcherDomainUserName": "mydomain.local\\login",
 "DirectorySearcherDomainUserPassword": "password"
  },
```

- Set the Express Conferences Bot API address in the key:

```
ExpressOptions.Uri:"https://my-CTS-domain.com/api/v1/Conference_bot/conferences/"
```

- Set the authentication type to be used when accessing the Express Conferences Bot API service in the key:

```
ExpressOptions.AuthenticationScheme:"StaticToken"
```

Note. StaticToken is the standard and recommended method of authentication using a static token.

- If this parameter takes the value "Basic", authentication is performed using a login/password (this method is non-standard and was used for versions below 1.2.0; its use is not recommended for higher versions).
- If the Customer requires Basic authentication for any reason, it is recommended to contact the development company for consultation.

Set the token value to be used when accessing the ExpressConferencesBotAPI service in the key:

```
ExpressOptions.Token:"Bearer dfgdf509usds0f9n0ud"
```

Note. This parameter must correspond to the bearer token value set during chat bot configuration. If authentication was not configured during chat bot setup, leave the configuration file parameters unchanged.

- Set the PostgreSQL database connection string in the key:

```
DatabaseOptions.Connection:"Host=127.0.0.1;Port=5432;Database=express_meeting_db;Username=db_express;Password=Pass1234;Pooling=true;Minimum Pool Size=50;Maximum Pool Size=100;Include Error Detail=True;"
```

Note. The parameters Username=db_express;Password=Pass1234 must correspond to the credentials created during PostgreSQL setup.

If a single AD domain is used in the infrastructure, proceed to the next step. If the Express CS is configured to synchronize users from multiple domains, and the IIS server hosting express-core-service receives requests from clients in other domains with trust relationships (including one-way trusts), configure the ADOptions section parameters.

Note. By default, this section is not set in the configuration file. If a multi-domain infrastructure is used, it must be created and configured manually. If you encounter difficulties with configuration, contact the developer for consultation.

- An array (list of domains in square brackets) to specify searching for users only in those domains. To include the current domain in the search, specify an empty string:

```
DirectorySearcherDomain: ["domain_name01", "domain_name02"]
```

- A string to specify the account name used for searching users in domains:

```
DirectorySearcherDomainUserName: "user"
```

- A string to specify the password of the account used for searching users in domains:

```
DirectorySearcherDomainUserPassword: "password"
```

Note. A detailed description of the ADOptions section in JSON format is provided in Appendix 5.

8. Run:

```
iisreset
```

9. Perform PostgreSQL database initialization with conference templates. In the command prompt, navigate to the folder containing express-core-service and run:

```
SET ASPNETCORE_ENVIRONMENT=Production

dotnet ExpressMeeting.CoreService.dll --environment Production --ef-migrate
```

10. If necessary, add exceptions to the antivirus for files in the following locations:
    - C:\IIS\core-service (the path where core-service was placed – the PhysicalPath specified in Chapter 4, Step 1, item 2);
    - C:\logs (if a different path was specified in Chapter 4, Step 2, item 2 then specify that path).

STEP 2. CONFIGURING SERVER-SIDE AND CLIENT-SIDE LOGGING FOR THE ADD-IN

To configure logging for the server-side and client-side parts of the add-in:

1. Grant FullControl permissions to Everyone on the server where cores-service is installed, for the folder c:/logs/core-service/[3] and its contents.
2. If necessary, change the path to the logs folder (see footnote to item 1 of the section "Step 2. Configuring Server-Side and Client-Side Logging for the Add-in").

STEP 3. DEPLOYING THE OUTLOOK ADD-IN ON CLIENT PCS

If the service was already deployed as part of a previous add-in version, then remove the previous version of the client software on the client machines and install the new version.

Important! After installing or updating the client part of the add-in on a PC with Windows OS, MS Outlook restart is required.

To deploy the Outlook add-in:

1. Run the ExpressMeeting.X.Y.Z.N.msi on the machine where Outlook is installed (X.Y.Z.N. is the release version).
2. After installation, in the ExpressMeeting.dll.config[4] file, set the server-side add-in service address in the key (replacemy-IIS-domain.com with the ip/dns name of the machine where express-core-service is deployed):

```
<add key="ExpressMeetingUrl" value="http://my-IIS-domain.com:7100/api/" />
```

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

Example: ExpressMeeting_v_3.1.0.2.msi ExpressMeetingUrl=http://IIS-name.ru:7100/api/

You can also use /qn for silent installation of the Outlook add-in (administrator rights are required).

3. If necessary, in the same ExpressMeeting.dll.config file, set the restriction mode for conference type selection. The drop-down list with the type selection

[3] This folder is intended for storing server-side and client-side logs. The path to the folder is specified in the appsettings.Production.json configuration file, in the key Serilog.WriteTo[Name="Logger"].Args.configureLogger.WriteTo[Name="File"].Args.path.

[4] The suggested path is c:\Program Files\Express\ExpressMeeting\ExpressMeeting.dll.config.

(Public/Corporate/Trusted)[5] 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, pass the DenyLinkType parameter in the command line.

Example: ExpressMeeting_v_3.1.0.2.msi DenyLinkType=true

4. If necessary, in the same ExpressMeeting.dll.config file, set the minimum password requirement level (possible values: Public, Trusts, Corporate). The password toggle 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, pass the LinkMinimumLevelRequirePassword parameter in the command line. Example: ExpressMeeting_v_3.1.0.2.msi LinkMinimumLevelRequirePassword =Corporate

5. If necessary, set the default conference creation type (possible values: Public, Trusts, Corporate). Meetings of the specified type will be created when clicking the "Create Conference" button. 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, pass the DefaultLinkType parameter in the command line.

Example: ExpressMeeting_v_3.1.0.2.msi DefaultLinkType=Corporate.

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

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

7. If it is necessary to enable sending diagnostic logs to the server during add-in installation, set the telemetryUrl parameter in the MSI package installation command line:

```
telemetryUrl:"http://my-IIS-domain.com:7100/api/log/json"
```

Example: ExpressMeeting_v_3.1.0.2.msi telemetryUrl="http://my-IIS-domain.com:7100/api/log/json" where my-IIS-domain.com — is the IP address or DNS name of the server where express-core-service is deployed

Important: if the telemetryUrl parameter is not specified during installation, the section with "Name": "Telemetry" is not created in the serilogSettings.json file, and sending logs to the server is disabled.

8. If necessary, add the following file locations to the antivirus exclusions on client PCs:
   - C:\Program Files\eXpress\ExpressMeeting (this path may differ for branded add-in versions);
   - C:\logs (if a different path was specified in Step 3, item 6, you should specify it here).

[5] More details on the types of conferences that can be created are available in the Express CS user guides.

If the service was already deployed as part of a previous add-in version, and the update was performed according to "Step 1. Deploying express-core-service in IIS", then:

- Ensure that the account used for the update has all necessary file system permissions.
- Replace the service files with the files from the distribution's IIS\template-service folder (except for the configuration file appsettings.Test.json. Do not replace this file! Otherwise, you will have to reconfigure the settings described in item 7).
- Rename the saved appsettings.Test.json file to appsettings.Production.json.
- On the IIS server, change the value of the ASPNETCORE_ENVIRONMENT parameter to Production (setting this value is described in item 6).

To deploy express-template-service in IIS:

1. Create an AppPool in IIS (Figure 8):

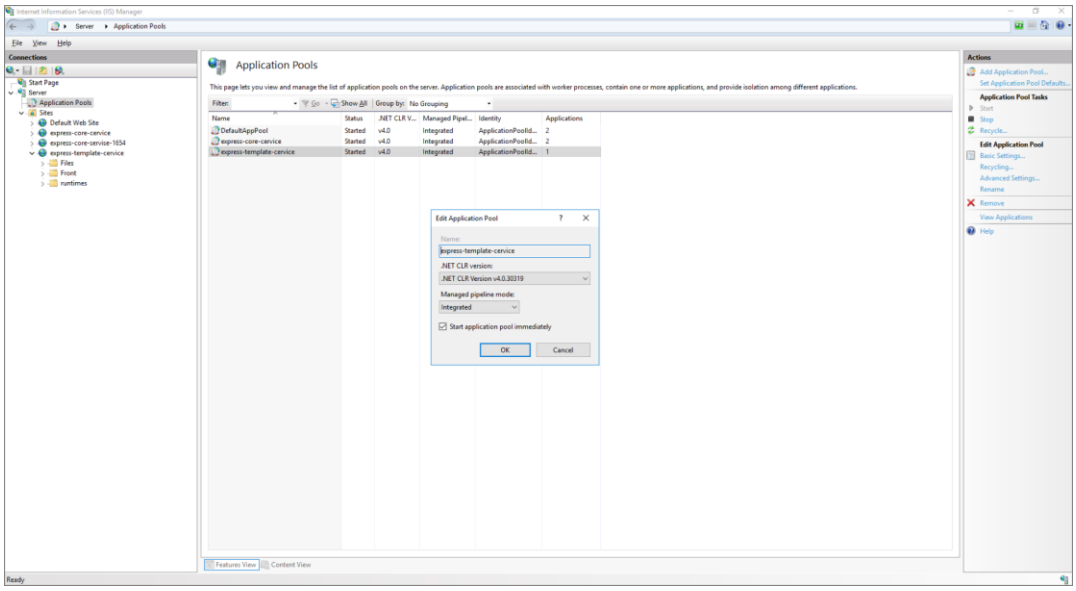

Figure 8

2. Create the express-template-service website.

   When creating, the path to the IIS\template-service folder must be specified in the PhysicalPath field.

3. In the Application pool field, select the pool created earlier.
4. In the Port field, specify the port that the client will use to communicate with the IIS server.
5. Navigate to "Home Node" → "Feature Delegation, Custom Site Delegation" (Figure 9). In the window that opens for the express-template-service website, grant read/write permissions for the Authentication – Anonymous parameter.

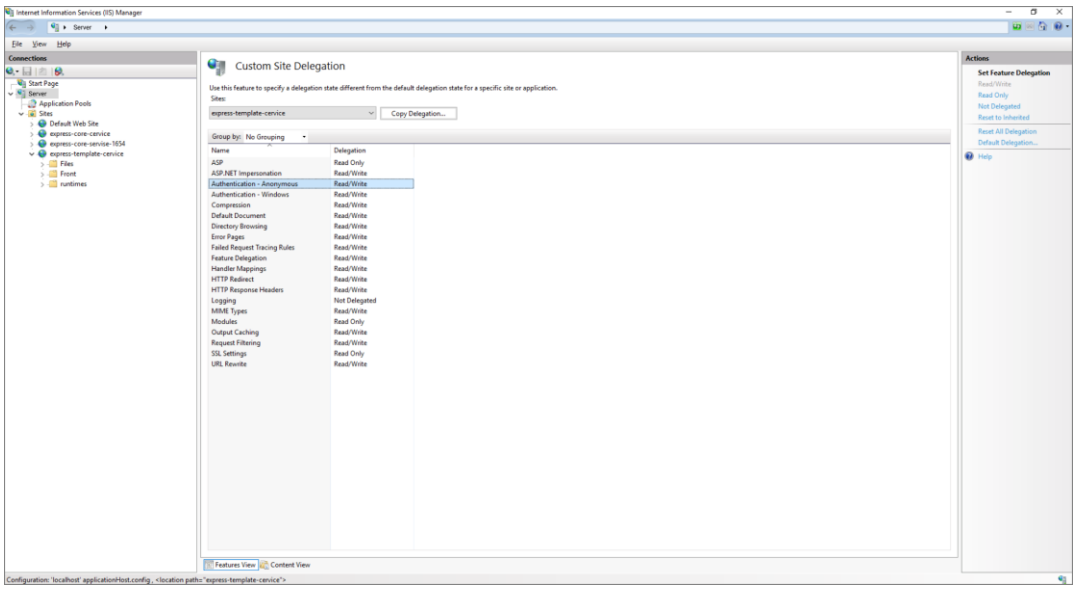

Figure 9

6. For the express-template-service website node, use the Configuration Editor to set the runtime environment via system.webServer/aspNetCore → environmentVariables From ApplicationHost.config (Figure 10). Fill in the parameters:

```
name: ASPNETCORE_ENVIRONMENT
value: Production
```

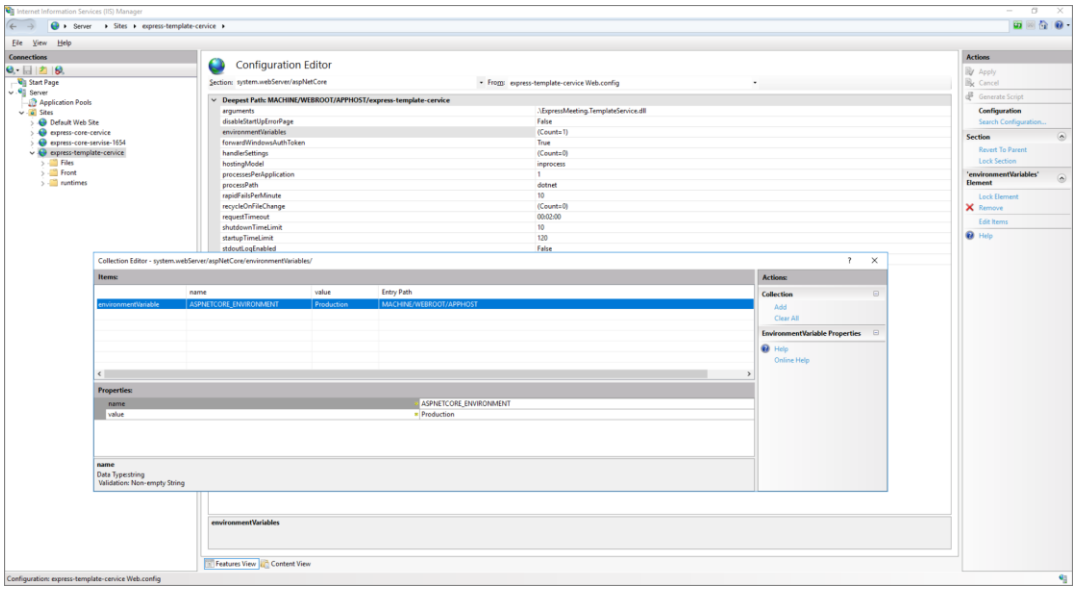

Figure 10

7. Edit the environment parameters in the appsettings.Production.json file[6]:
   - set the PostgreSQL DB connection string in the DatabaseOptions section, Connection parameter:

```
Connection:"Host=127.0.0.1;Port=5432;Database=express_meeting_db;
Username=db_express;Password=Pass1234;Pooling=true;Minimum Pool
Size=50;Maximum Pool Size=100;Include Error Detail=True;"
```

Note. The connection string parameters Username=db_express;Password=Pass1234 must match the credentials that were created during PostgreSQL configuration. The database is the same one used by express-core-service.

[6] Предполагаемый путь IIS\template-service\appsettings.Production.json.

- set the settings for generating the JWT authentication token (used when accessing service methods) in the JWTOptions section with the following parameter values:

```
Key:"JFRINXV0LGIhLXlEVzlYdXlae316UU0zJzR9WVUtIXA"
Issuer:"Issuer"
Audience:"Audience"
ExpiredMinutes:120
```

Note. The Key parameter contains an arbitrary set of characters used to generate the key that the token will be signed with. The ExpiredMinutes parameter sets the token lifetime in minutes. It is acceptable not to change the parameters unless required by company information security requirements.

8. Run the command:

```
iisreset
```

9. If the website is successfully deployed, navigating to /front (example: http://localhost:7200/front) in a browser should open the login page for the add-in template administration system (Figure 11).

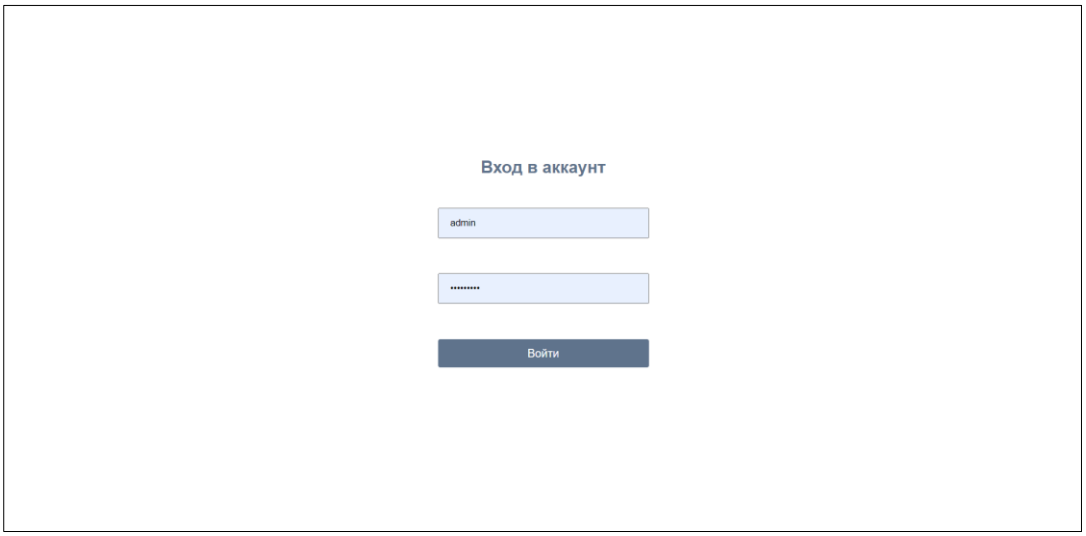

Figure 11

The first login is performed using the admin account with an empty password.

10. If necessary, add files from the following locations to the antivirus exclusions:
    - C:\IIS\template-service (the path where the template service was deployed – the PhysicalPath that was specified in Chapter 4, Step 4, item 2);
    - C:\logs (if a different path was specified in Chapter 4, Step 4, item 2 you should specify it here).

STEP 5. CONFIGURING LOGGING FOR EXPRESS-TEMPLATE-SERVICE

To configure logging for express-template-service:

1. Grant FullControl permissions to Everyone on the server where express-template-service is installed, for the folder c:/logs/template-service/ and its contents.
2. If necessary, change the path to the logs folder (see footnote to item 1 of the section "Step 2. Configuring Server-Side and Client-Side Logging for the Add-in").

Chapter 5

ADMINISTRATOR WEB INTERFACE

This chapter describes the main sections and procedures for working with the administrator web interface.

AUTHORIZATION IN THE 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 11).

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 11).
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 12).

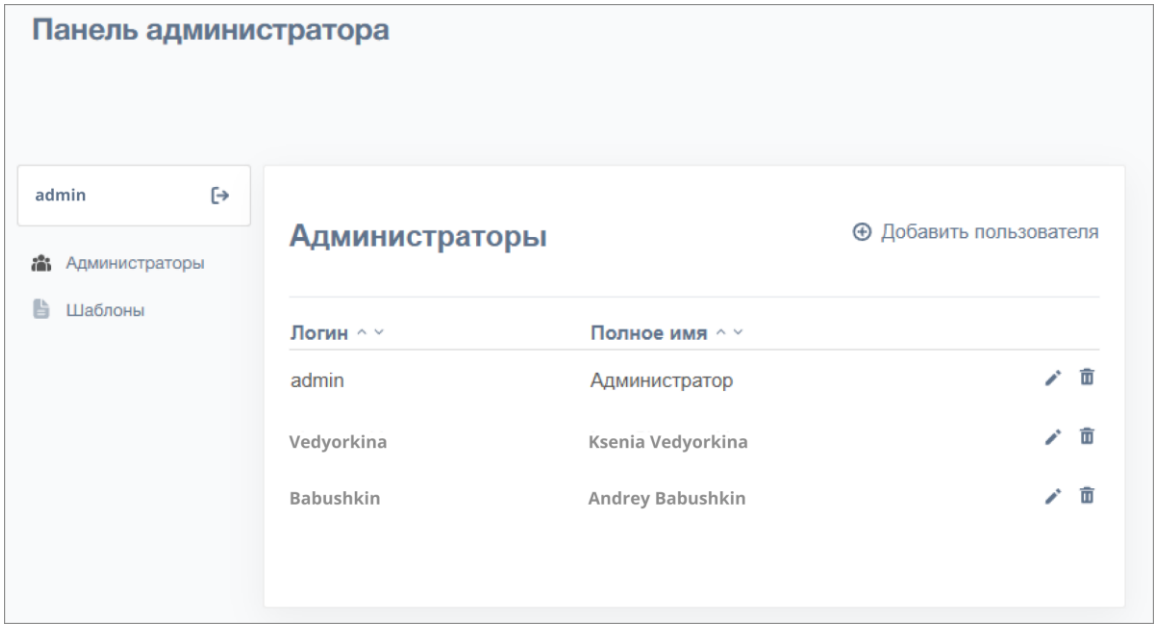

Figure 12. "Administrators" Section Interface

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 13).

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

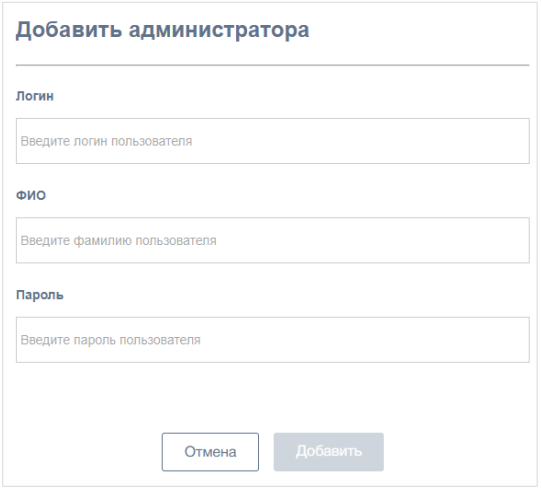

Figure 13. Adding an administrator

To edit administrator data:

1. Select an administrator from the list (Figure 12) and click the "Edit" button.

   The administrator data editing form will open (Figure 14):

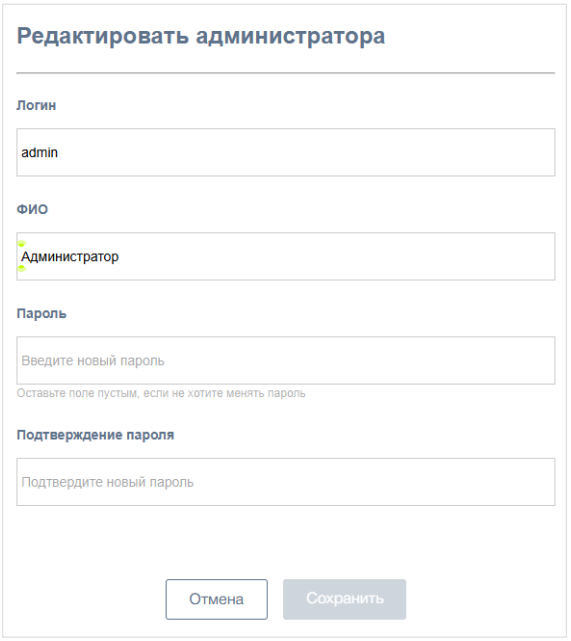

Figure 14. Making changes to administrator data

2. Make the necessary changes and click the "Save button".

To delete an administrator, select the administrator from the list (Figure 12) and click the button.

In the “Templates” section (Figure 15), the administrator can create, edit (including in HTML mode), and delete invitation email templates.

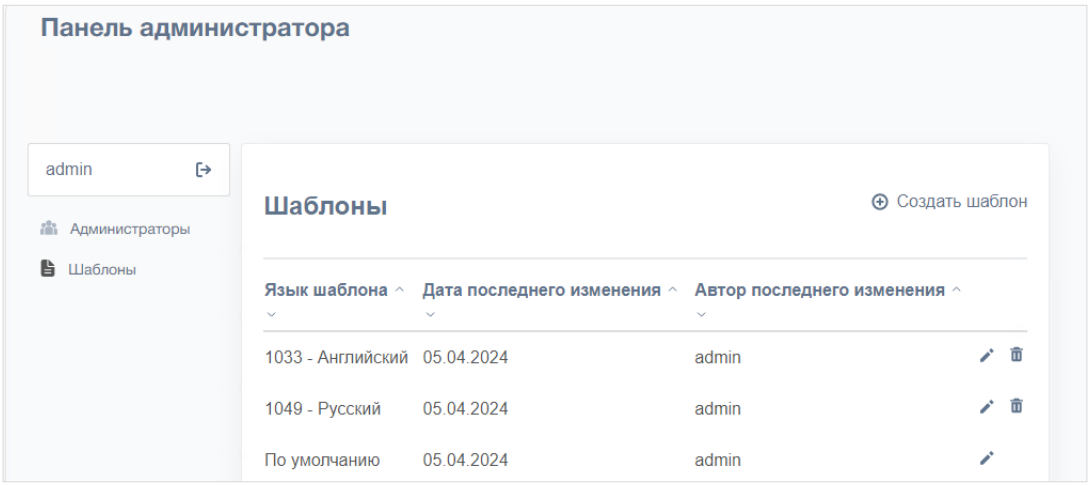

Figure 15. The “Templates” section

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 16).

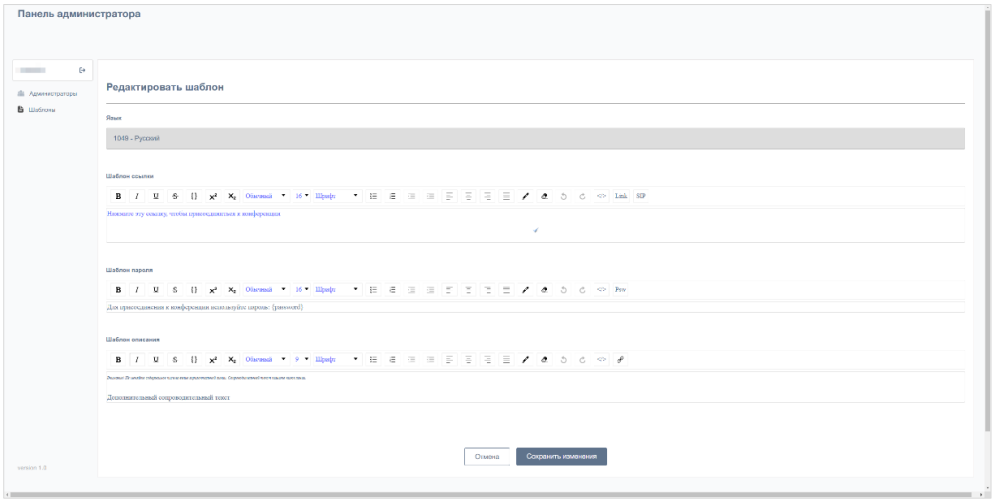

Figure 16. Visual Template Editor

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

To edit a template:

1. Select a template from the list (Figure 15) and click the button.

   The visual template editor will open (Figure 16).

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

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

To switch to the HTML Editor, click the "</>" in visual template editor menu. The editor window will look as follows (Figure 17):

Figure 17. 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="
rvf39+zr6bXI4qizwufdz5WhsmSt/5rC+r3Ezm1zeJiSjmum8tzm9bvZ/6bB5a6qpn+t5dvV
zZK88+v8/7vg/7DJ4P/99V5gY8zMzObm5ofD/6zQ/3Fua8fX69fm+vDy9OPi4czh/4SXrJLC
/////+7u7Wmt/87f9oG2/5Oku5mZmf///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAACH5BAUUADEALAAAAAANAAwAAAZTwJhwSIwhDsUixZEkWhLNYeQVjYUY
BIVKSCKeGIOCymCISTATISsCu5RKW1VAkHKBUBDToRxTcUYNGhsdFR8GW0IqJS0ZDyIrh0kq
CwBIVR4eTUEAOw==">
```


In the invitation email, this will be displayed as follows (Figure 18):

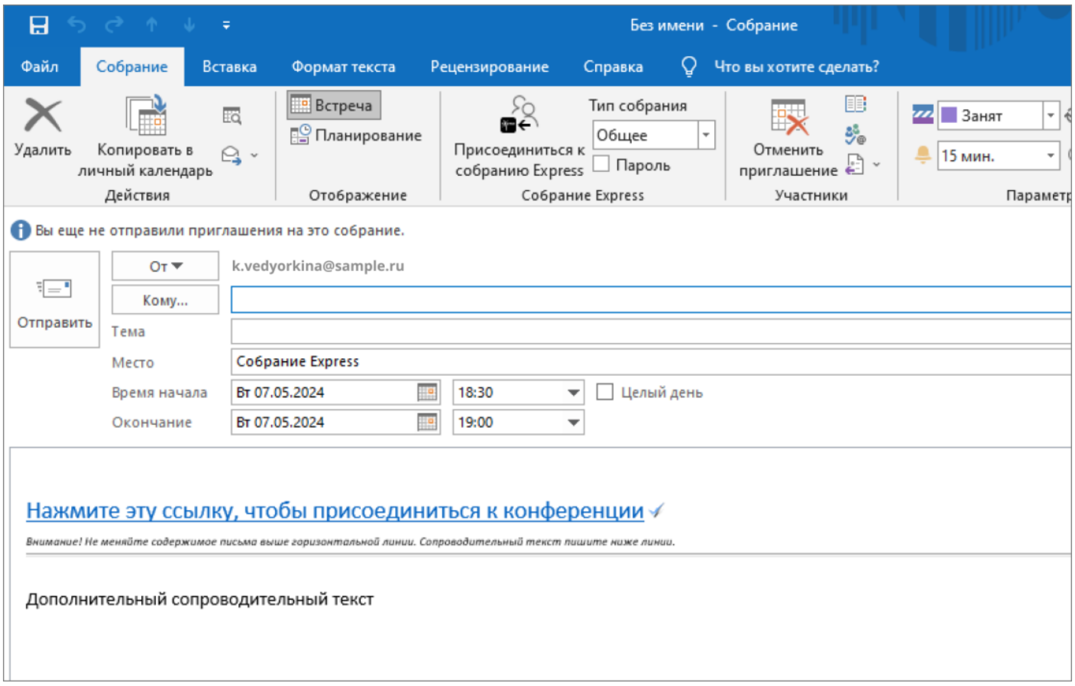

Figure 18. Result of working in the HTML Editor

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

OUTLOOK ADD-IN TROUBLESHOOTING

This section does not contain instructions for correcting identified malfunctions but helps in conducting diagnostics.

Principle of the Outlook Add-in's Operation:

1. Upon clicking the "Create Meeting" button in Outlook, a request is formed and sent to the IIS server.
2. Authentication and authorization are performed on the IIS server.
3. If authentication and authorization are successful, IIS passes the email template from the DB to the Outlook add-in and sends a request to the chatbot to create a template for the future conference.
4. The Conference Notifier Bot creates a template for the future conference in eXpress.
5. The client part of the Outlook add-in creates an email in Outlook from the template, inserting a link to the future conference.
6. The user specifies attendees, date, time, and name of the future conference by filling in the corresponding fields in the Outlook email.
7. After the user clicks the "Send" button, the Outlook add-in forms and sends a request to the IIS server to modify the previously created conference, taking into account the parameters filled in by the user.
8. If authentication and authorization are successful, the IIS server sends a request to the chatbot to modify the template for the future conference, taking into account the parameters filled in by the user.
9. The chatbot updates the conference parameters in eXpress.

The procedure for diagnosing a malfunctioning Outlook add-in is 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 "Enabling Conference Notifier Bot API Without Password";
   - If chatbot security **was** configured, see items 7-9 of the section "Securing Conference Notifier Bot".

STEP 2. CHECKING CONFERENCE NOTIFIER BOT ACCESSIBILITY FROM THE IIS SERVER

Next, it is recommended to check the accessibility of the Conference Notifier Bot from the IIS server.

To check the accessibility of the Conference Notifier Bot from the IIS server:

1. Connect via RDP or another method to the IIS server where the express-core-service application is installed.
2. In PowerShell ISE, run the diagnostic script from Appendix 1, Diagnostic Script No. 1, editing the variable values (usernames, link, conference date and time).
3. If the script ran with errors, analyze and eliminate their causes (most likely a problem with network access or authentication), then run the script again until it runs without errors.
4. If the script ran without errors and returned a conference ID, which can be found in the eXpress Corporate Server (CTS) administration console, the Conference Notifier Bot is considered accessible from the IIS server, proceed to the next step.

STEP 3. CHECKING IIS SERVER ACCESSIBILITY FROM THE CLIENT WORKSTATION

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

To check the accessibility of the IIS server from the client workstation:

1. Connect via RDP to the client workstation where Microsoft Outlook and the Outlook add-in are installed.
2. In PowerShell ISE, run the diagnostic script from Appendix 2, Diagnostic Script No. 2, editing the variable values (Outlook user account, link to IIS).
3. If the script ran with errors, analyze and eliminate their causes (most likely a problem with IIS and express-core-service application settings, lack of network access, or authentication and authorization issues), then run the script again until it runs without errors.
4. If the script ran without errors and returned a conference ID, which can be found in the eXpress Corporate Server (CTS) administration console, the Conference Notifier Bot is considered accessible from the IIS server, proceed to the next step.

STEP 4. GENERAL OUTLOOK ADD-IN DIAGNOSTICS

If the previous 3 steps were successful, but the **Outlook add-in still does not work**, check the correctness of the settings in the **client software configuration file** (see the section "Step 3. Deploying the Outlook Add-in").

If the client software configuration file is correct, review the **log files** from the client applications, described in the section "Step 3. Deploying the Outlook Add-in" of this instruction, and the server log files, described in the section "Step 2. Configuring Server-Side and Client-Side Logging for the Add-in", eliminate the causes of the errors, and if necessary, contact the 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.

```
# This script is intended to check the availability of the Conference Notifier Bot from the express-core-service IIS server.
# It is only meaningful to run this script after the Conference Notifier Bot's availability has been successfully checked via CURL from the Express server. See the latest version of the Express Outlook Add-in installation instructions.
# Run this from the IIS server that ensures the functioning of the Express Outlook Add-in (preferably in PowerShell ISE, as both the script body and the result will be visible on the screenshot).
# If this script runs successfully, a conference with a CallID will be created in Express (this script will output the CallID).
# You can find this conference in the Express administrator console using the CallID.
# The successful execution of this script indicates that the Conference Notifier Bot is available from the IIS server.
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/json")
$headers.Add("Authorization", "Bearer 3bfef52b32685598d1f73301dd3c0431") # enter your bearer token here if set, and if not set, leave it as is or comment it out.
# Below, fill in the conference name, members, creator, and date with time (the date must not be earlier than the current date). It is better to change the conference name on repeat runs.
$body = @"
{`"name`":null,`"members`":[`"User001@mydomain.com`"],`"admins`":[`"User001@mydomain.com`"],`"creator`":`"User001@mydomain.com`",`"start_at`":`"0001-01-01T00:00:00`",`"end_at`":`"0001-01-01T00:00:00`",`"link`":{`"link_type`":`"public`",`"access_code`":null}}
"@
$response = Invoke-RestMethod 'https://my-CTS-domain.com/api/v1/conference_Bot/conferences/' -Method 'POST' -Headers $headers -Body $body
$response | ConvertTo-Json
$callId = $response.result.call_id

$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/json")
$headers.Add("Authorization", "Bearer 3bfef52b32685598d1f73301dd3c0431") # enter your bearer token here if set, and if not set, leave it as is or comment it out

# Below, fill in the conference name, members, creator, and date with time (the date must not be earlier than the current date). It is better to change the conference name on repeat runs.
$body = @"
{`"name`":`"Change name`",`"members`":[`"User001@mydomain.com`",`"User002@mydomain.com`"],`"admins`":[`"User001@mydomain.com`"],`"actor`":`"User001@mydomain.com`",`"start_at`":`"2023-12-31T12:30:00+03:00`",`"end_at`":`"2023-12-31T13:00:00+03:00`",`"link`":{`"link_type`":null,`"access_code`":null}}
"@
$response = Invoke-RestMethod "https://my-CTS-domain.com/api/v1/conference_Bot/conferences/$callId" -Method 'POST' -Headers $headers -Body $body
$response | ConvertTo-Json
```

Appendix 2

DIAGNOSTIC SCRIPT NO. 2

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

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

```
# This script is intended to check the availability of the Conference Notifier Bot+IIS express-core-service bundle from the client (PC where Outlook is installed).
# It is only meaningful to run this script after the Conference Notifier Bot's availability has been successfully checked via CURL from the Express server (see the latest version of the Express Outlook Add-in installation instructions) and with the test_from_IIS_PC_to_Express_PC_x_x.ps1 script from IIS express-core-service.
# Run this from the client (PC where Outlook is installed) (preferably in PowerShell ISE, as both the script body and the result will be visible on the screenshot).
# If this script runs successfully, a conference with a CallID will be created in Express (this script will output the CallID).
# You can find this conference in the Express administrator console using the CallID.
# The successful execution of this script indicates that the Conference Notifier Bot is available from the IIS server and the IIS express-core-service is configured correctly.
$creds = Get-Credential # AD OUTLOOK User Account
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/json")

# Fill in the creator below:
$body = @"
{`"Creator`":`"Express_User@mydomain.com`",`"LanguageId`":1049,`"isRecurrent`":false,`"`$type`":`"MeetingCreateRequest`"}
"@

# Enter the IIS express-core-service address below; the same link must be in the client add-in settings.

$response = Invoke-RestMethod 'http://my-iis-domain.com:7100/api/meetings' -Method 'POST' -Headers $headers -Body $body -Credential $creds
$response | ConvertTo-Json
```

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 (IIS Server)	7100/TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with express-core-service001 on the IIS server (standard network port specified).
2	express-core-service (IIS Server)	CTS Front (CTS Single)	443/TCP	Interaction of express-core-service on the IIS server with the Conference Notifier Bot API on the CTS Front (CTS Single) server.
3	Template Administrator (Client PC)	express-template-service (IIS Server)	7200/TCP	Interaction of the Template Administrator with express-template-service001 on the IIS server via the web interface (standard network port specified).
4	express-core-service (IIS Server)	PostgreSQL DB (DB Server)	5432/TCP	Interaction of express-core-service001 on the IIS server with the PostgreSQL DB on the DB server (standard network port specified).
5	express-template-service (IIS Server)	PostgreSQL DB (DB Server)	5432/TCP	Interaction of express-template-service on the IIS server with the PostgreSQL DB on the DB server (standard network port specified).
Note. The table specifies the standard network ports for interaction between client PCs, the IIS server, and domain controllers within Active Directory for seamless authentication.				

Table 5. Multiple CTS without ETS

No.	Source	Destination	Port and Protocol	Interaction Description
No.	Source	Destination	Port and Protocol	Interaction Description
1	Internal User on CTS01 (Client PC)	express-core-service001 (IIS Server)	7100/TCP	Interaction of the Outlook add-in on the user's PC on CTS01 with express-core-service001 on the IIS server (standard network port specified).
2	Internal User on CTS02 (Client PC)	express-core-service002 (IIS Server)	7300/TCP	Interaction of the Outlook add-in on the user's PC on CTS02 with express-core-service002 on the IIS server (standard network port specified).
3	Template Administrator (Client PC)	express-template-service (IIS Server)	7200/TCP	Interaction of the Template Administrator with express-template-service on the IIS server via the web interface (standard network port specified).
4	express-core-service001 (IIS Server)	PostgreSQL DB (DB Server)	5432/TCP	Interaction of express-core-service001 on the IIS server with the PostgreSQL DB on the DB server (standard network port specified).
5	express-template-service (IIS Server)	PostgreSQL DB (DB Server)	5432/TCP	Interaction of express-template-service on the IIS server with the PostgreSQL DB on the DB server (standard network port specified).
6	express-core-service002 (IIS Server)	PostgreSQL DB (DB Server)	5432/TCP	Interaction of express-core-service002 on the IIS server with the PostgreSQL DB on the DB server (standard network port specified).
7	express-core-service001 (IIS Server)	CTS Front (CTS Single)01	443/TCP	Interaction of express-core-service001 on the IIS server with the Conference Notifier Bot API on the CTS Front (CTS Single)01 server.
Note. The table specifies the standard network ports for interaction between client PCs, the IIS server, and domain controllers within Active Directory for seamless authentication.				

Table 6. Multiple CTS with ETS

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

Notes:

- The table specifies the standard network ports for interaction between client PCs, the IIS server, and domain controllers within Active Directory for seamless authentication.
- Network interaction between the EXPRESS ETS servers and EXPRESS CTS servers is detailed in “Express. Communications System. Administrator's Guide. Volume 1. Installation” https://express.ms/admin_guide_install.pdf.

Appendix 4

COMON ISSUES

Issue 1

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 connection string in the appsettings.Production.json file[7] or the DatabaseOptions.Connection parameter:

- first try setting Minimum Pool Size=0;
- if the previous step does not solve the problem, set Pooling=False.

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

Issue 2

Description: An error occurs when launching the Configuration Editor in IIS during express-core-service configuration (Chapter 4, item 6, page **Ошибка! Закладка не определена.**) (Figure 19):

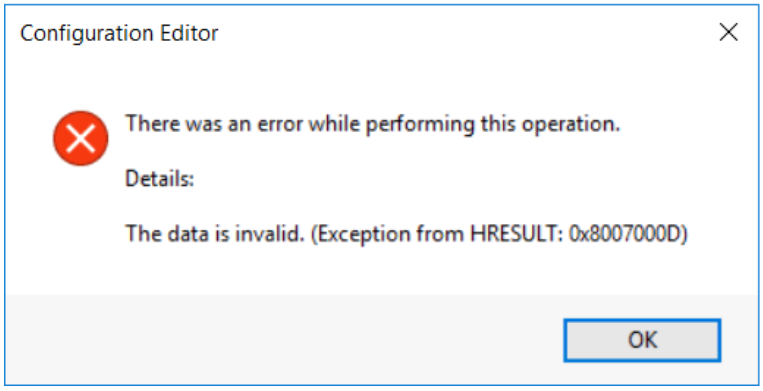

Figure 19

Possible Solution: Install the following package: https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-aspnetcore-8.0.8-windows-hosting-bundle-installer.

Important! Dotnet packages must be installed after IIS installation.

Issue 3

Description: An error occurs when clicking the add-in button in MS Outlook (Figure 20):

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

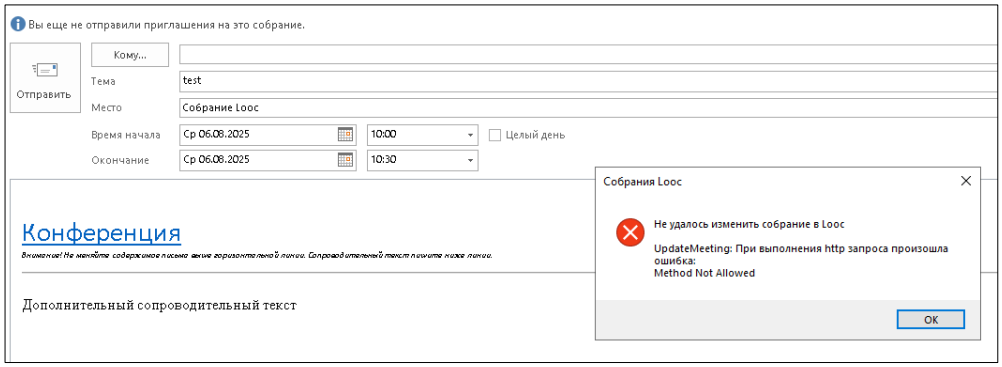

Figure 20

The following message appears in the logs:

```
[ERR] <1> Error - Update Meeting: Failed to modify meeting in Initiator: ExpressMeeting.Core.Domain.Exceptions.HttpExpressMeetingDomainException. Error: Method Not Allowed.
```

Possible Solution: Disable WebDav in IIS.

Appendix 5

ADOPTIONS SECTION IN APPSETTINGS.PRODUCTION.JSON

Example with a single domain:

```
"ADOptions": {
     "DirectorySearcherDomain": ["domain_name01"],
        "DirectorySearcherDomainUserName": "user",
        "DirectorySearcherDomainUserPassword": "pass",
    }
```

Example with multiple domains:

```
"ADOptions": {
     "DirectorySearcherDomain": ["domain_name01","domain_name02"],
        "DirectorySearcherDomainUserName": "user",
        "DirectorySearcherDomainUserPassword": "pass",
    }
```

Example with multiple domains and including the current domain in the search:

```
"ADOptions": {
     "DirectorySearcherDomain": ["domain_name01","domain_name02",""],
        "DirectorySearcherDomainUserName": "user",
        "DirectorySearcherDomainUserPassword": "pass",
    }
```

If the values for DirectorySearcherDomainUserName and DirectorySearcherDomainUserPassword are not specified, the IIS User Account will be used to connect to the domain(s).