



Enghouse
Interactive

Installation and Configuration

TE API

Trio Enterprise 8.2



Last update: October 15, 2020



COPYRIGHT NOTICE:

No part of this document may be reproduced, distributed, stored in a retrieval system or translated into any language, including but not limited to electronic, mechanical, magnetic, optical, photocopying, manual or other form of reproductions, without the prior written permission of Enghouse Interactive. For additional copies of this document, please contact Enghouse Interactive.

This document contains proprietary information of Enghouse Interactive and is protected by copyright law. All Rights Reserved.

The information in this documentation is subject to change without notice and is provided "as is". Enghouse Interactive reserves the right to revise the contents or withdraw access to them at any time. Enghouse Interactive makes no warranty of any kind to this document and shall in no event be liable for errors herein.

Copyright © Enghouse Interactive AB 2020

Document history

| Version | Date | Author | Note |
|---------|------------|---------------------------------|--|
| PA1 | 2018-11-07 | Jaan Kaja / Anna Lundkvist | First version for TE 8.0 created based on the TE 7.1 manual "CC API Installation and Configuration". |
| PA2 | 2019-01-02 | Anna Lundkvist | Added the test tool. |
| PA3 | 2019-08-14 | Anna Lundkvist | Updated the images of the test tool. |
| PA4 | 2019-11-07 | Nonis Malamas | First version for TE 8.1. Added appendix with field identities. |
| PA5 | 2019-12-17 | Anna Lundkvist | Added AccessProfileId and profileCategoryId to Appendix B. |
| PA6 | 2020-01-16 | Mikael Norberg | Added referrals to Appendix B. |
| PA7 | 2020-02-28 | Mats Behre / Anna Lundkvist | Removed the CD API module and modified DMZ installation info. |
| PA8 | 2020-03-25 | Richard Ekefjård | Updates for TE 8.1 FP1: CC backend now works as the CD backend. |
| PA9 | 2020-05-15 | Mikael Norberg / Anna Lundkvist | Updates for TE 8.1 FP1: Added fields in Appendix B. |
| PA10 | 2020-07-01 | Anna Lundkvist | First version for TE 8.2. |
| PA11 | 2020-09-04 | Anna Lundkvist | Added fields in Appendix B. |
| PA12 | 2020-10-15 | Anna Lundkvist | Updates on how to configure the CC backend server port and how to use the test tool. |

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | About this document | 1 |
| 1.2 | Related documents | 1 |
| 1.3 | Terminology and acronyms..... | 1 |
| 1.4 | About the TE API | 2 |
| 1.4.1 | About the CC API | 2 |
| 1.4.2 | About the CD API | 2 |
| 1.5 | System overview | 3 |
| 1.6 | Failover | 3 |
| 1.7 | Usage | 4 |
| 1.7.1 | The web help | 4 |
| 1.7.2 | Example projects | 4 |
| 2 | Prerequisites | 5 |
| 2.1 | General prerequisites..... | 5 |
| 2.2 | Required information..... | 5 |
| 2.2.1 | Parameters for creating tasks..... | 5 |
| 3 | Installation and configuration | 7 |
| 3.1 | Adding the API licences | 7 |
| 3.2 | Creating an API account | 8 |
| 3.3 | Configuring the CC API..... | 9 |
| 3.3.1 | The CC API in EMC | 9 |
| 3.3.2 | Task server licence..... | 10 |
| 3.3.3 | Task routing | 11 |
| 3.4 | Configuring the CD API..... | 13 |
| 4 | Test..... | 14 |
| 4.1 | Basic test..... | 14 |
| 4.2 | Using the test tool | 15 |
| 5 | Support | 18 |
| 5.1 | Logging | 18 |
| 5.1.1 | REST API frontend | 18 |
| 5.1.2 | CC API backend | 18 |
| 5.1.3 | CD API backend | 19 |
| 5.1.4 | Additional log files | 20 |

| | |
|---|---------------|
| Appendix A – DMZ server | 21 |
| Introduction..... | 21 |
| Proxy server rules | 22 |
| Example – Configuration using IIS | 22 |
| Appendix B – Field identities..... | 35 |
| Subscriber data | 35 |
| Organisation data | 40 |
| Security..... | 42 |

1 Introduction

1.1 About this document

This document describes the TE API briefly, as well as how to install, configure and support it on the Trio Enterprise server.

In addition, this document describes which information that must be provided to the developer that will utilize the TE API, including information about how to find the web based help with details about how to use the API.

Appendix A describes how to set up a DMZ server for the TE API. Appendix B describes the subscriber, organisation and security fields that are accessible through the API.

1.2 Related documents

The following documents are referred to in this manual:

| Reference | Document |
|-----------|--|
| [1] | Contact Center Configuration CC_ContactCenter Configuration.pdf |
| [2] | User's Manual Trio Administrator |

1.3 Terminology and acronyms

| Abbreviation | Explanation |
|--------------|-----------------------------------|
| TE | Trio Enterprise |
| CC | Contact Center |
| CD | Company Directory |
| EMC | Enterprise Management Center |
| API | Application Programming Interface |
| REST | Representational State Transfer |

1.4 About the TE API

The TE API is a REST API that can be used by third party developers to integrate with a Contact Center and/or the Company Directory in Trio Enterprise. It is divided into two different parts, the CC API and the CD API, that use the same frontend application.

1.4.1 About the CC API

The CC API provides programmatic access to reading and managing configuration data about agents and services, as well as reading real time contact center data. In addition, it enables the possibility to create, modify and delete tasks in the contact center queue.

The CC API can for example be used for wallboard applications and to display contact center status such as current waiting time and queue length on a website. The possibility to create tasks in the queue can for example be used to let an alarm system automatically create a task for distribution to an agent in the contact center.

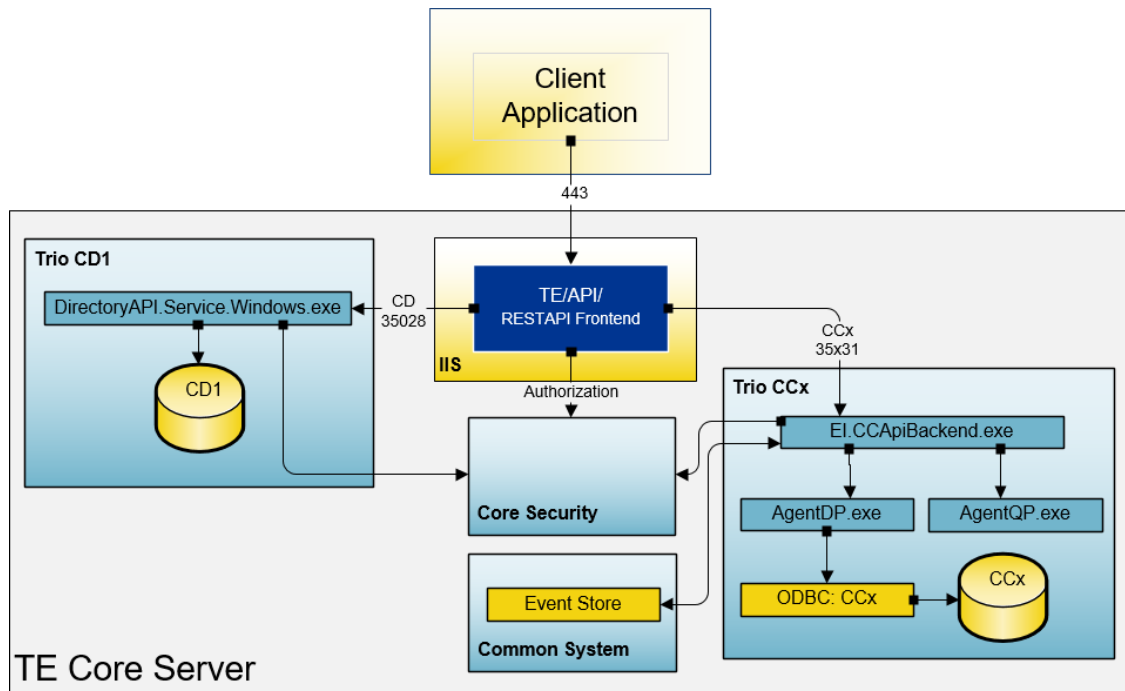
1.4.2 About the CD API

The CD API provides programmatic access to managing and reading subscriber information in the Company Directory as well as reading organisation data. The most commonly used fields can be accessed using the CD API. A subscriber can be added as an extension, a user or an attendant/agent user. Administrative permissions and agent settings in the CC cannot be added through the CD API. See [Appendix B – Field identities](#) or details about which fields that can be managed and read through the CD API.

The CD API can for example be used for provisioning of subscribers and users in Trio Enterprise.

1.5 System overview

The CC API backend runs on the server with the *Contact Center Core* role and the CD API backend runs on the server with the *Company Directory* role. These roles must be assigned to the same server. The CC API and the CD API backends use the same frontend application in IIS. The CC API is a module that must be enabled in EMC, while the CD API is always active.



If the TE API is to be accessible externally see [Appendix A – DMZ server](#).

1.6 Failover

The API supplies information about both the primary and the secondary server. In cases when the system runs in failover mode and the backend processes on the primary server is still running, calls to the primary server are redirected to the secondary server and vice versa. If needed, the redirection can be deactivated.

Statistical data in the CC will not be accurate in failover mode. For example, some figures will show data starting from the time when the switchover was made.

CD data cannot be added or updated on the secondary server.

1.7 Usage

1.7.1 The web help

The usage of the TE API is described in a web help that is available through the API once it is enabled on the TE core server. Browse to the following URL to access the help. Replace *<hostname>* with the hostname of the Trio Enterprise core server or the proxy server.

[http\(s\)://<hostname>/te/api/help/](http(s)://<hostname>/te/api/help/)

1.7.2 Example projects

Together with the software for Trio Enterprise, open source projects are available as examples of how the API can be used.

The projects are found in the Trio Enterprise release. The project for the CC API is located in the folder **Doc_API\TERestApiClientSample** and the project for the CD API is located in the folder **Doc_API\API.Tool.Windows**.

2 Prerequisites

2.1 General prerequisites

The following is required for all integrations with the TE API:

- Licence keys for the CC API and/or the CD API, acquired from Enghouse Interactive. These keys must be distributed to the developer that will utilize the API.
- If the TE API should be accessible externally, a reverse proxy should be configured according to the instructions in [Appendix A – DMZ server](#).
 - If a DMZ server is configured for Trio Administrator Web, that server may be used for the TE API without further configuration.
- The roles **Contact Center Core** and the **Company Directory** roles must run on the same server.

Additional requirement if the CC API should add, modify or delete tasks:

- A *Task Server* licence for the Contact Center.

2.2 Required information

The developer that will utilize the TE API must be provided with the following information:

- The Trio Enterprise licence key for the API (CC and/or CD) that is to be utilized, acquired from Enghouse Interactive.
- Credentials for an API account in Trio Enterprise. See [Creating an API account](#) on page 8 for more information.

Additional information that is needed if the CC API should add, modify or delete tasks:

- Information about CC, service id and other parameters as described below.

2.2.1 Parameters for creating tasks

When a task is created by the CC API, it is possible to set a number of parameters for the case in Trio Enterprise. The following parameters can be set when creating a task using the CC API.

| API Parameter | TE Call Parameter | Description |
|---------------|-------------------|--|
| ccid | | The id of the Contact Center. E.g., the ccid of CC1 is 1. This field is mandatory. |
| ServiceId | sid | The service id where the task initially should be placed. This field is mandatory. |

| API Parameter | TE Call Parameter | Description |
|-------------------------|------------------------------------|--|
| Header | subj | A short description of what this task entails. The text will be shown in the task message box when the agent receives the task and possibly also in the job information pane in Agent Client. |
| Description | body | A detailed description of this task. The text will be shown in the task message box when the agent receives the task. |
| RouteCategory | routecat | This value may be used to route the tasks in Interaction Studio. |
| Category | ex.calltag | A category property that will be shown in the statistics. |
| Force | force | A flag to disregard the normal rules for when a case may be added to a service queue, e.g. if it is closed or unstaffed. |
| Creator | origin | A context string used to identify who created the task in the system log files. |
| Priority | pclass | The ID of a task priority to decide how fast the task will advance through the queue. The IDs of the default set of priority classes are: 1 - low (1 queue point) 2 - normal (2 queue points) 3 - high (3 queue points) 4 - very high (10 queue points) |
| CustomParameters | c.xxx d.xxx tc.xxx td.xxx | Four types of custom parameters can be added to a task: <ul style="list-style-type: none"> • <i>Parameters</i> that will get the prefix c.. • <i>PersistentParameters</i> that will get the prefix d.. • <i>TakeCallParameters</i> that will get the prefix tc.. • <i>PersistentTakeCallParameters</i> that will get the prefix td.. <p>The prefixes are added by the CC API, so the parameters should be sent to the API without the prefix.</p> <p>It is important that the custom parameters that are used is added in Trio Administrator with the correct prefix.</p> <p>See reference [1] Contact Center Configuration for more information about custom parameters and their prefixes.</p> |
| Event | event extpoponly | If a screen pop event should be executed when the task is accepted by an agent, this parameter defines its name and whether the default task dialog in Agent Client should be suppressed. |
| CustomerId | cust | The customer number. |
| GPSCoordinates | gps | GPS coordinates for the case that for example can be used to display a map for the agent. |
| PreferredAgents | | A list of IDs of the preferred agents. |
| PreferredAgentsWaitTime | | For how long, in seconds, the task will wait for the preferred agents before it is assigned to another available agent. |

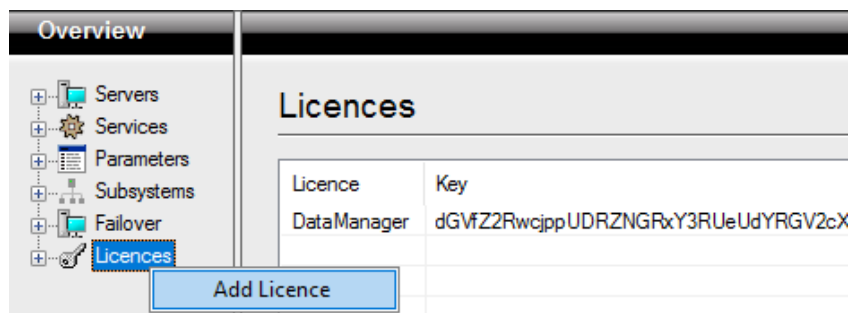
3 Installation and configuration

3.1 Adding the API licences

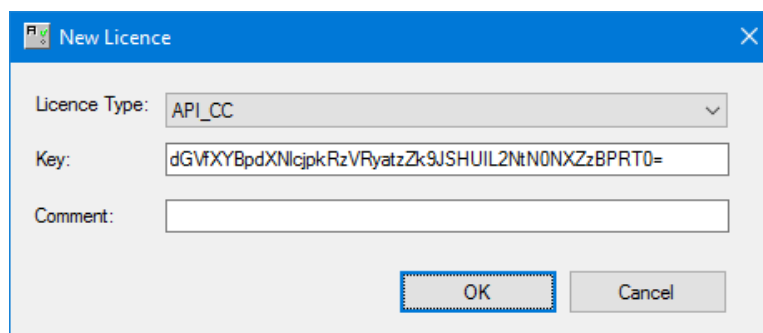
There are separate licences for the CC API and the CD API. Both licence keys should be added in EMC.

Do the following for each of the licence keys that is to be added:

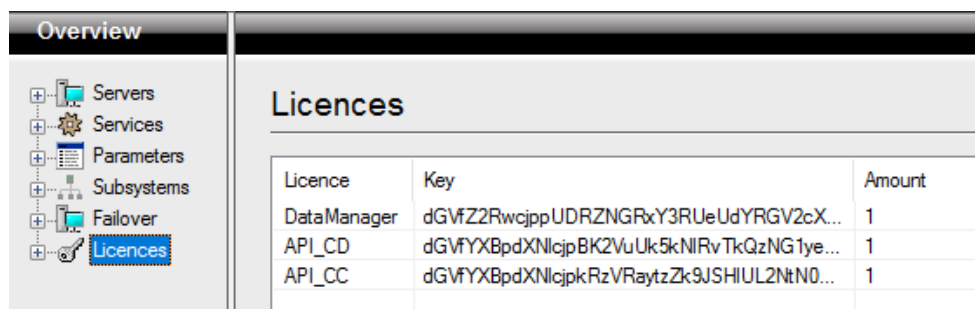
1. Right click **Licences** in the tree structure and select **Add Licence**.



2. Select the licence type for the API that is to be installed (**API_CC** or **API_CD**) and enter the licence key.



3. Click **OK**.
4. Check that the amount of the newly added licence is 1. This means that the licence has been successfully added.



5. Activate the configuration.

3.2 Creating an API account

To enable a third party application to connect to and retrieve data from Trio Enterprise, it must be authenticated with a user account of the type *API account*. The account must also have a permission profile that gives permissions to view the requested data.

1. Create a new user in Trio Administrator.
2. On the **Security** page, select the subscriber type **API Account** and select a permission profile.

Note: The built-in profile *API* have full permissions to perform all operations available through the API. Additional permission profiles of the category **API Account** can be created if there is need to limit the permissions of the account. See reference [\[2\] User's Manual Trio Administrator](#) for more information about permission profiles.

3. Set a username and a password for the user. These credentials must be distributed to the developer that will utilize the TE API.

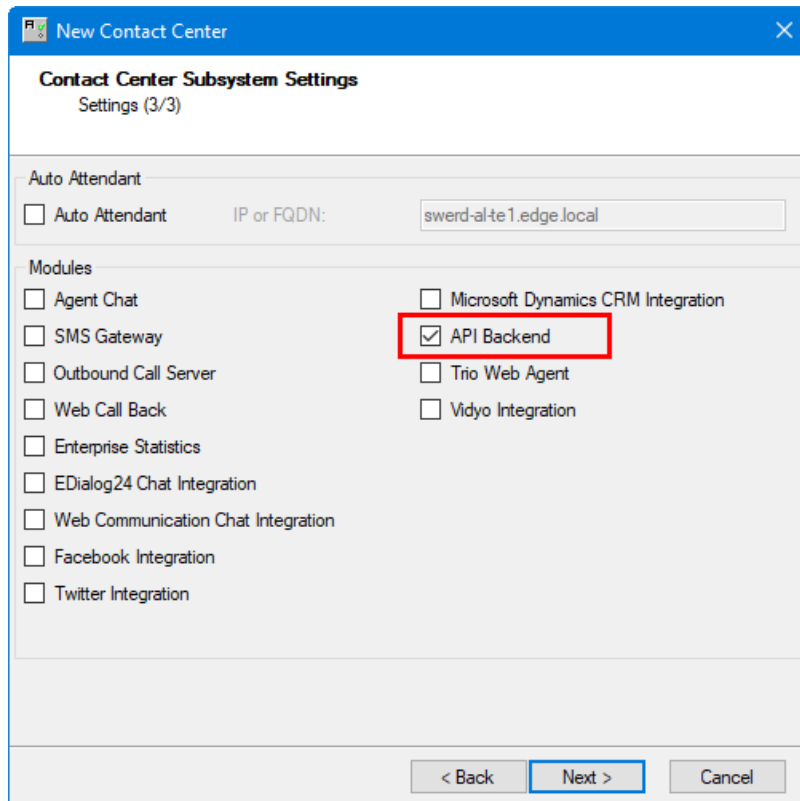
The screenshot shows the 'Add Subscriber' dialog box with the 'Security' tab selected. The 'Type of Subscriber' is set to 'API Account'. The 'Authentication' section shows 'Login Name' as 'teapiuser' and 'Login Password' as a masked password. The 'Privileges' section shows the 'Profile' set to 'API account'. The 'User account status' section shows 'The user account will be unlocked' checked and 'Locked' unchecked. The 'Add Subscriber' button is highlighted.

3.3 Configuring the CC API

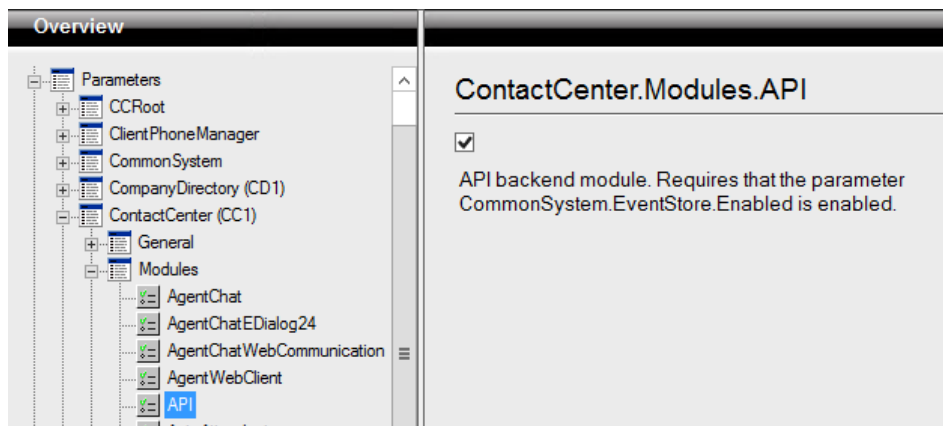
3.3.1 The CC API in EMC

The CC API is a module in the Contact Center subsystem and all files needed for it are installed when installing the Trio Enterprise core server. However, the module must be enabled.

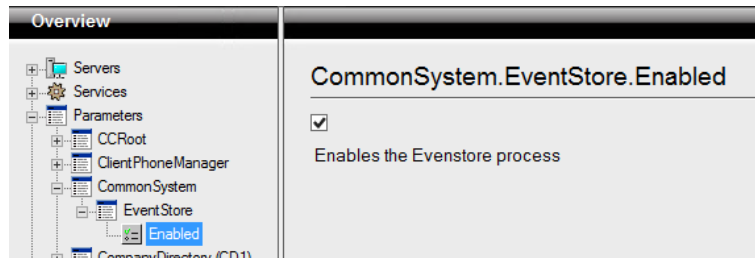
1. To enable the CC API, select **API Backend** when adding a Contact Center subsystem in EMC.



It is also possible to enable the module in an already existing CC in EMC. This is done by activating the module **API** under **Parameters** → **ContactCenter** → **Modules**.

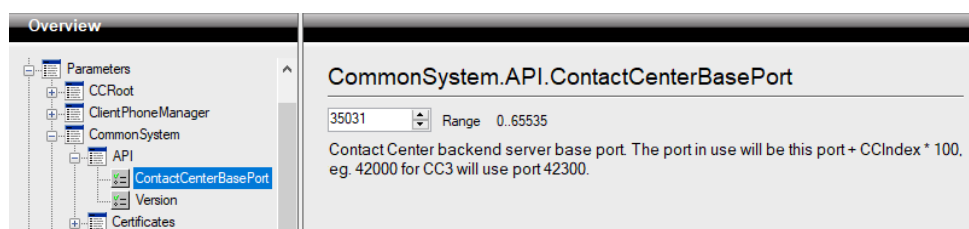


- Under **CommonSystem** → **EventStore**, add and enable the parameter **Enabled**.



- In most systems, the default port 35x31 for the backend server port can be used. This means port 35131 for CC1 etc. However, the port can be changed if needed.

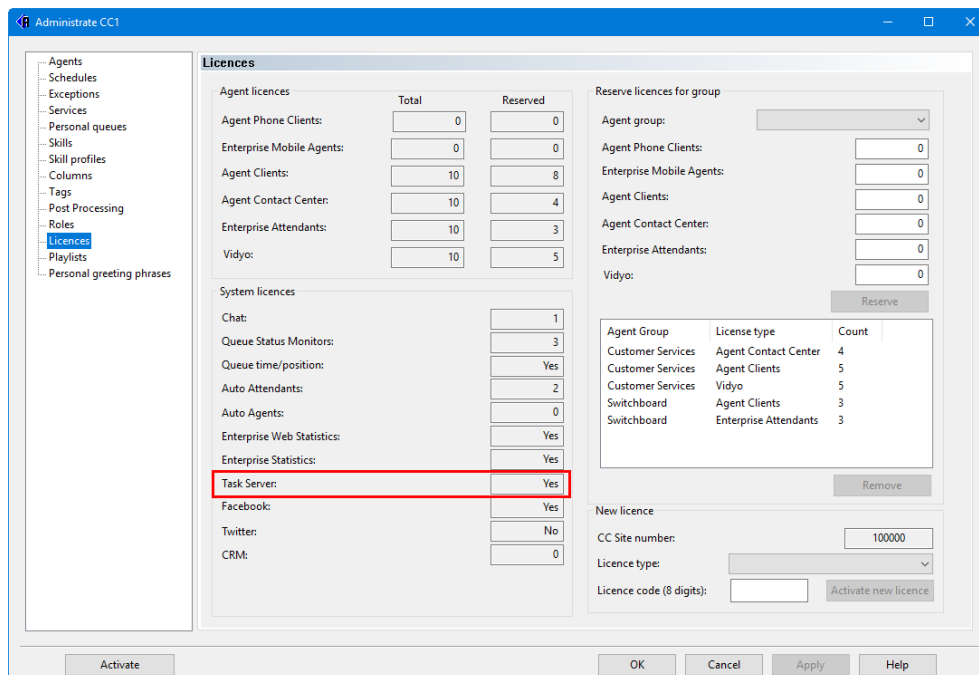
To do this, add the parameter **CommonSystem.API.ContactCenterBasePort** and set it to the desired value. This parameter changes the port for all CCs, which means that the base port number must be entered, e.g. 42000 if port 42100 is to be used for CC1 and 42200 for CC2 etc.



- Activate the configuration and restart the services **Trio CCx** and **Trio Common Service**.

3.3.2 Task server licence

If the CC API should be used to add, modify or delete tasks, a licence is required in the Contact Center. Make sure that the **Task Server** licence is activated for the Contact Center in Trio Administrator.



3.3.3 Task routing

If **rotecat** is set when creating a task, it may be used to route the task in Interaction Studio.

The value of **rotecat** is used in the routing table in Interaction Studio to send the cases to the wanted entrance. In this example, all cases where **rotecat** is set to **task** will be routed to the entrance called **Tasks**.

| Routing | | | | |
|--------------------|-------------------|---------------------|----------|--|
| Case routing table | | | | |
| Field | Value | CC/Entrance | Language | |
| Email | *@triolab16.local | Entrance - Email | English | |
| Routing Category | cmb | Entrance - CMB | English | |
| Routing Category | chat | Entrance - Chat | English | |
| Routing Category | task | Entrance - Tasks | English | |
| C-No. | 6160..6163 | Entrance - Services | English | |
| C-No. | 6168..6169 | Entrance - VG/VM | English | |
| C-No. | * | Entrance - Default | English | |

Using the value of **RouteCategory** to route a task to an entrance.

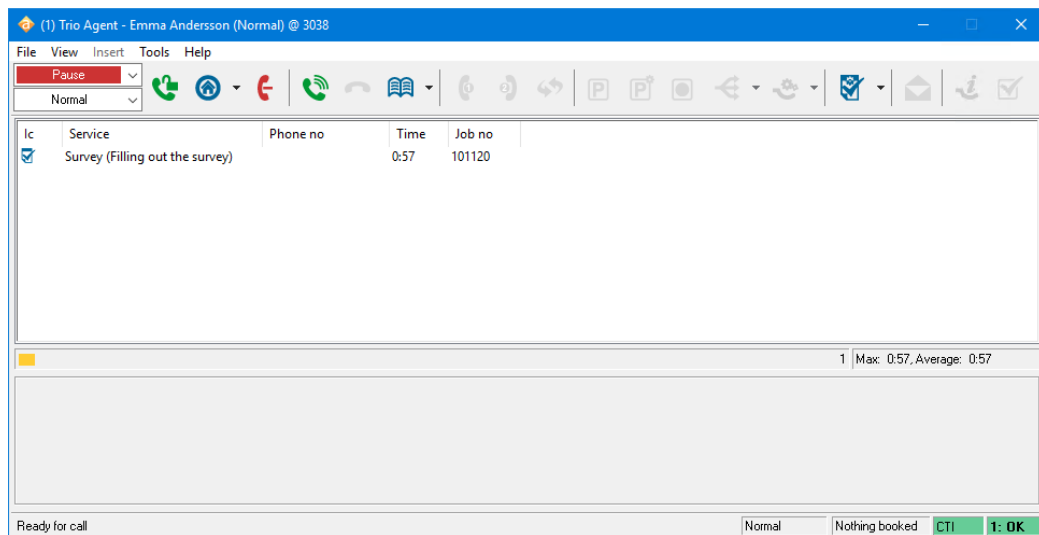
In the entrance, the verb **Evaluate Call Parameter** can be used to evaluate other call parameters to route each task to the correct exit. In this example, the call parameter **subj** is evaluated and based on its content, the tasks are distributed to different services.

| Overview | Evaluate Call Parameter |
|----------|---|
| | <div>Evaluate Call Parameter</div> <div>Call parameter to evaluate subj</div> <div>Extract a subset from the call parameter to evaluate Startpos <input type="text"/> Number of characters <input type="text"/></div> <div>If the call parameter contains car use exit 1</div> <div>If the call parameter contains survey use exit 2</div> <div>If the call parameter use exit</div> <div>If the call parameter use exit</div> |

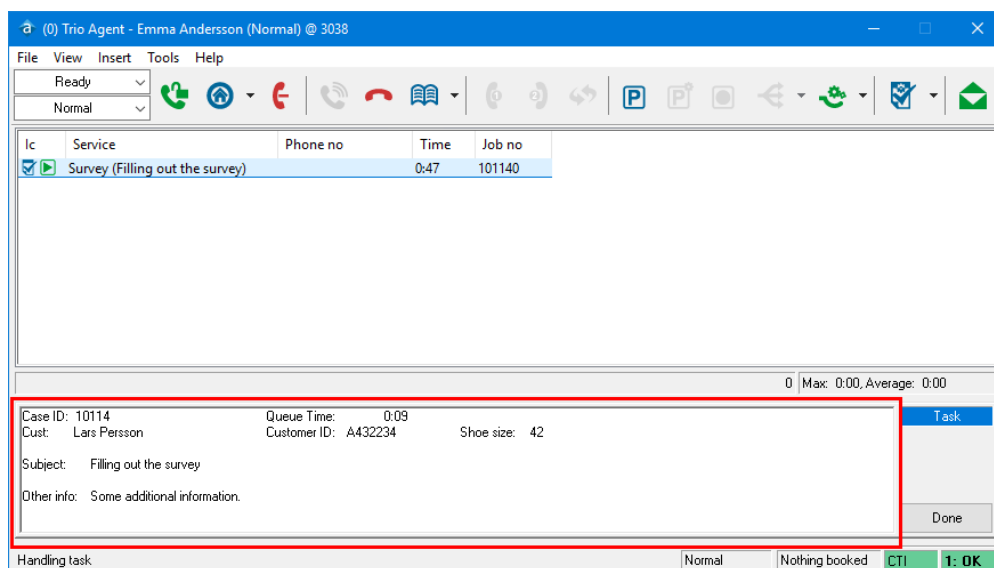
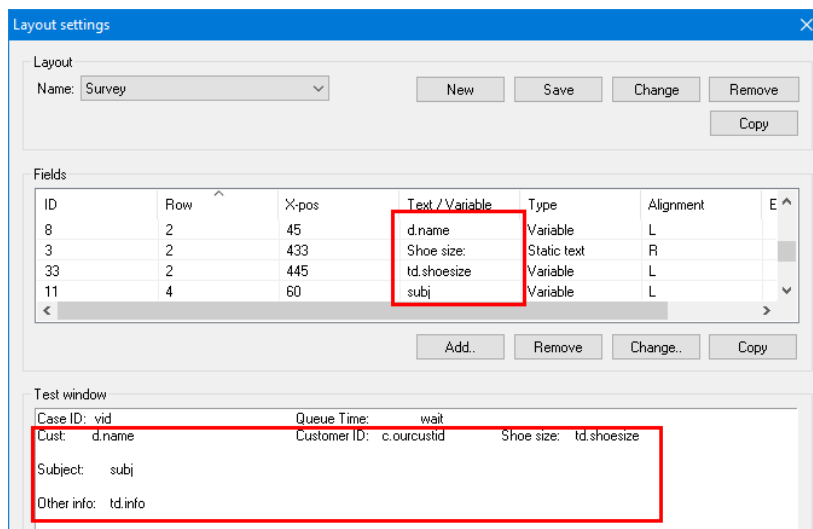
The **Evaluate Call Parameter** looks for the word “survey” in the call parameter **subj**.

If the word “survey” is found in the value of the call parameter **subj**, the task will be routed to exit number 2 and placed in queue for the service called **Survey**.

| Overview | Service |
|----------|---|
| | <div>Service</div> <div>Service ID 8 - Survey</div> <div><input type="checkbox"/> Include redirect information</div> <div><input type="checkbox"/> Use calling number (A-no) as customer ID</div> <div>Retrieve name information for all call parties from Company Directory CD1</div> |



To present relevant information for the agent when a task is accepted in Agent Client, a customized layout can be created. In this example, the layout shows custom parameters that were set when the task was created.



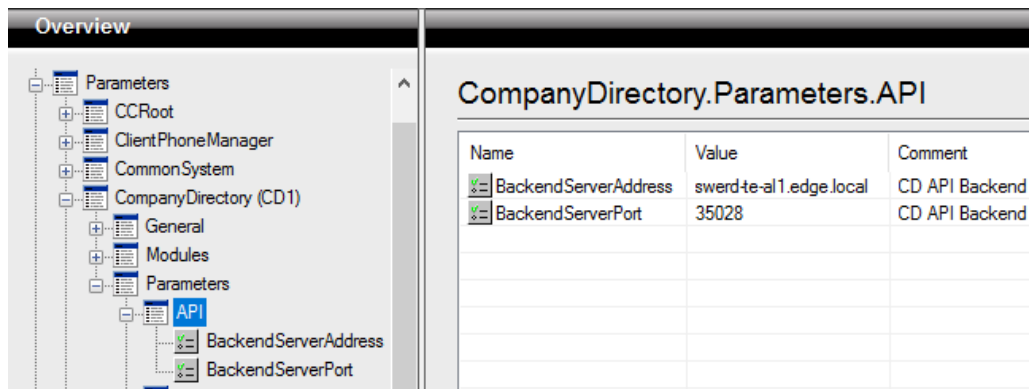
3.4 Configuring the CD API

The CD API is a part of the Company Directory subsystem and all files needed for it are installed when installing the Trio Enterprise core server.

The CD API is always active and most systems can use the default settings. However, there are some parameters that may be added and configured for the CD API if needed.

1. In EMC, go to **CompanyDirectory** → **Parameters** → **API**.
2. Add and configure the following parameters as needed.

| Parameter | Description |
|----------------------|---|
| BackendServerAddress | The FQDN or IP address of the server with the <i>Company Directory</i> role. It will be used by the API frontend application to connect to the CD API backend process. In most systems, the default value <i>localhost</i> can be used. Note: This parameter can also be set in the <i>Add Company Directory</i> wizard. |
| BackendServerPort | The port that CD API backend listens on. Default value is 35028. |



The screenshot shows the EMC configuration interface. On the left is a tree view under 'Parameters' with a sub-tree for 'CompanyDirectory (CD1)' containing 'General', 'Modules', 'Parameters', and 'API'. The 'API' node is selected. On the right is the 'CompanyDirectory.Parameters.API' configuration page, which contains a table with the following data:

| Name | Value | Comment |
|----------------------|-------------------------|----------------|
| BackendServerAddress | swerd-te-al1.edge.local | CD API Backend |
| BackendServerPort | 35028 | CD API Backend |
| | | |
| | | |
| | | |
| | | |

3. Activate the configuration and restart the service **Trio CD1**.

4 Test

4.1 Basic test

To test that the TE API is functioning, start a web browser and enter the following URI. Replace *<hostname>* with the hostname of the Trio Enterprise core server.

[http\(s\)://<hostname>/te/api](http(s)://<hostname>/te/api)

The result should be a page with relative links to other REST resources:

- *Self* is the link to this page.
- *Help* is the link to the web help for the API.
- *ContactCenters* is the link to list all CCs in the system. This requires authentication.
- *Directory* is the link to the CD management. This requires authentication.
- *Login* is the link used for authentication and login.

Example:

```
{
  "links": [
    {
      "rel": "self",
      "desc": "",
      "method": "GET",
      "href": "/te/api/"
    }
  ],
  "data": [
    {
      "links": [
        {
          "rel": "Help",
          "desc": "Trio Enterprise REST API Help",
          "method": "GET",
          "href": "/te/api/help"
        },
        {
          "rel": "ContactCenters",
          "desc": "Get all Contact Centers",
          "method": "GET",
          "href": "/te/api/cc"
        },
        {
          "rel": "Directory",
          "desc": "Directory management",
          "method": "GET",
          "href": "/te/api/directory"
        }
      ],

```

```

    {
      "rel": "Login",
      "desc": "Get access and refresh tokens",
      "method": "POST",
      "href": "/te/api/login"
    },
    {
      "rel": "RefreshLogin",
      "desc": "Renew access and refresh tokens",
      "method": "POST",
      "href": "/te/api/refreshlogin"
    }
  ],
  "binaryVersion": "8.0.2.0",
  "failoverMode": false,
  "primaryServer": "swe-teapi-core.edge.local",
  "secondaryServer": "swe-teapi-fo.edge.local"
}
]
}

```

4.2 Using the test tool

A tool for testing the API key and the API user, as well as the CD API, is available on the Trio Enterprise server. The tool is called **API.Tool.Windows.exe** and is located in **X:\TE\ProgramFiles\<te_patch_level>\bin\CD\Server\CDApiBackend\<fp_patch_level>**.

Testing login

This command verifies that the API key and the API user credentials are correct. This command can be used for testing both the CC and CD API.

1. Start a command window on the Trio Enterprise server.
2. Change to the directory of the test tool:
`X:\TE\ProgramFiles\<te_patch_level>\bin\CD\Server\CDApiBackend\<fp_patch_level>`
3. Run the following command:

```
API.Tool.Windows.exe login APIKey=<CC/CD key> user=<API user>
pwd=<API psw> host=<FQDN>
```

Replace <CC/CD key> with the license key for the CC or CD API.

Replace <API user> with the username of the API user account.

Replace <API psw> with the password of the API user account.

Replace <FQDN> with the FQDN of the Trio Enterprise server.

Example with a correct result:

```
Administrator: C:\windows\system32\cmd.exe - API.Tool.Windows.exe login APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMbJF...  
D:\TE\ProgramFiles\8.2.11\bin\CD\Server\CDApiBackend\8.2.11>API.Tool.Windows.exe login APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMbJF...  
XNlcjoyaHNFM1YvMlhMbJF...V08zT0pGTRZMPT0= user=teapiuser pwd=Enghouse20 host=swerd-al-te1.edge.local  
Token: d24c4b710f993c6ad02cef8eb031bc74  
RefreshToken: b3f9583405d87e201e395917fadfa211  
Elapsed time ms: 211  
> _
```

Testing search

When the login command has been run successfully, as described in [Testing login](#) above, the following command can be run to verify that the API key is correctly entered in EMC and that the API user has permissions to view subscribers.

1. Run the following command:

```
search data=lastName=<subscriber name>
```

Replace <subscriber lastname> with the last name of a subscriber in the CD.

Example with a correct result:

```
Administrator: C:\windows\system32\cmd.exe - API.Tool.Windows.exe login APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMbJF...  
D:\TE\ProgramFiles\8.2.11\bin\CD\Server\CDApiBackend\8.2.11>API.Tool.Windows.exe login APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMbJF...  
XNlcjoyaHNFM1YvMlhMbJF...V08zT0pGTRZMPT0= user=teapiuser pwd=Enghouse20 host=swerd-al-te1.edge.local  
Token: d24c4b710f993c6ad02cef8eb031bc74  
RefreshToken: b3f9583405d87e201e395917fadfa211  
Elapsed time ms: 211  
> search data=lastName=Andersson  
Subscriber search: Emma Andersson  
Subscriber search: Ext 6166 pbxid 2  
Elapsed time ms: 43  
> _
```

Testing login and search with a single command

This command verifies that the API key is correct, that it is correctly entered in EMC and that the API user exists and has permissions to view subscribers.

1. Start a command window on the Trio Enterprise server.
2. Change to the directory of the test tool:
X:\TE\ProgramFiles\<te_patch_level>\bin\CD\Server\CDApiBackend
\<fp_patch_level>.
3. Run the following command:

```
API.Tool.Windows.exe search APIKey=<CD key> user=<API user>  
pwd=<API psw> host=<FQDN> data=lastName=<subscriber name>
```

Replace <CD key> with the license key for the CC or CD API.

Replace <API user> with the username of the API user account.

Replace <API psw> with the password of the API user account.

Replace <FQDN> with the FQDN of the Trio Enterprise server.

Replace <subscriber name> with the last name of a subscriber in the CD.

Example with a correct result:

```
Administrator: C:\windows\system32\cmd.exe - API.Tool.Windows.exe search APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMb...
D:\TE\ProgramFiles\8.2.11\bin\CD\Server\CDApiBackend\8.2.11>API.Tool.Windows.exe search APIKey=dGVfYXBpdXNlcjoyaHNFM1YvMlhMb...
dXNlcjoyaHNFM1YvMlhMbJFzV08zT0pGTRZMPT0= user=teapiuser pwd=Enghouse20 host=swerd-al-te1.edge.local dat
a=lastName=Andersson
Subscriber search: Emma Andersson
Subscriber search: Ext 6166 pbxid 2
Elapsed time ms: 290
>
```

Additional tests

If needed, other commands for managing subscribers are available in the test tool. Run the following command to view information about all available commands:

```
API.Tool.Windows.exe help
```

Or if already logged in to the test tool:

```
help
```

```
Administrator: C:\windows\system32\cmd.exe - API.Tool.Windows.exe help
D:\TE\ProgramFiles\8.2.11\bin\CD\Server\CDApiBackend\8.2.11>API.Tool.Windows.exe help
login host=<IP|FQDN> APIKey=a1c3ab34 user=abc pwd=****
refresh host=<IP|FQDN> APIKey=a1c3ab34 refreshToken=<your refresh token>
create [token=<your token> | APIKey=a1c3ab34 user=abc pwd=****] host=<IP|FQDN> data=firstName=abcd,...
update [token=<your token> | APIKey=a1c3ab34 user=abc pwd=****] host=<IP|FQDN> userId=<userId> data=firstName=abcd,...
delete [token=<your token> | APIKey=a1c3ab34 user=abc pwd=****] host=<IP|FQDN> userId=<userId> data=firstName=abcd,...
search [token=<your token> | APIKey=a1c3ab34 user=abc pwd=****] host=<IP|FQDN> data=firstName=abcd,... | data=?firstName
=abcd&lastName=xyz&...
get [token=<your token> | APIKey=a1c3ab34 user=abc pwd=****] host=<IP|FQDN> userId=<userId>
Within the tool you can set all parameters as from the cmd line. Just a return repeats the last cmd. Do CTRL-C or quit
to exit the tool
Elapsed time ms: 22
>
```

5 Support

5.1 Logging

The TE API consists of several process that produce log files in different locations.

5.1.1 REST API frontend

The REST API Frontend, which is hosted by IIS, logs to the file
X:TE\ProgramData\CommonSystem\log\Api.log.

Example from the log when using the test URI <http://<hostname>/te/api>:

```
...
20:14.46.41-811: T= 62 : Logsettings created
20:14.46.41-811: T= 62 : APIConfig created
20:14.46.41-827: T= 62 : FrontEndRequest: (10.104.218.101) http://swe-ccapi-
test.edge.local/te/api/
20:14.46.41-827: T= 62 : Api: Returning OK
...
```

5.1.2 CC API backend

The CC API backend process *EI.CCApiBackend.exe* logs to the file
X:TE\ProgramData\CC1\log\CCApiBackend.log.

Example of a typical start-up log:

```
20:14.07.53-138: T= 1 : EI.CCApiBackend version 7.1.18.1 running.
20:14.07.53-278: T= 1 : Running in subsystem CC1
20:14.07.53-997: T= 1 : GetParUtil: Using OAM configuration 13
20:14.07.54-263: T= 1 : CmdSockConnection.StartMaintenance (QP0)
state=DISCONNECTED server=swe-ccapi-test:35101
20:14.07.54-263: T= 1 : QPConnManager: Added QP for CC1 on swe-ccapi-test
20:14.07.54-284: T= 1 : DbAccessor - Initializing TDL for CC API Backend...
20:14.07.54-378: T= 6 : CmdSockConnection.OnConnected (QP0)
20:14.07.57-696: T= 1 : Config changed: -1->13
20:14.07.58-103: T= 1 : DbAccessor - Found sub system: CC1
20:14.08.00-266: T= 1 : DbAccessor - *** Service: sid=1 name=Default
20:14.08.00-266: T= 1 : DbAccessor - *** Service: sid=2 name=Sales
20:14.08.00-266: T= 1 : DbAccessor - *** Service: sid=3 name=Support
20:14.08.00-266: T= 1 : DbAccessor - *** Service: sid=4 name=Logistics
20:14.08.00-266: T= 1 : DbAccessor - Initializing CC Database Model for CC
API Backend...
20:14.08.00-427: T= 1 : ServiceManager - Starting RPCServer instance: 0
20:14.08.00-443: T= 1 : ServiceManager - Starting RPCServer instance: 1
20:14.08.00-443: T= 11 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-443: T= 14 : RPCServer.Run - Awaiting RPC requests...
```

```

20:14.08.00-443: T= 1 : ServiceManager - Starting RPCServer instance: 2
20:14.08.00-459: T= 1 : ServiceManager - Starting RPCServer instance: 3
20:14.08.00-459: T= 17 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-459: T= 20 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-459: T= 1 : ServiceManager - Starting RPCServer instance: 4
20:14.08.00-459: T= 23 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-459: T= 1 : ServiceManager - Starting RPCServer instance: 5
20:14.08.00-521: T= 26 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-521: T= 1 : ServiceManager - Starting RPCServer instance: 6
20:14.08.00-521: T= 29 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-521: T= 1 : ServiceManager - Starting RPCServer instance: 7
20:14.08.00-521: T= 32 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-521: T= 1 : ServiceManager - Starting RPCServer instance: 8
20:14.08.00-521: T= 35 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-537: T= 1 : ServiceManager - Starting RPCServer instance: 9
20:14.08.00-569: T= 38 : RPCServer.Run - Awaiting RPC requests...
20:14.08.00-646: T= 5 : EventStore.Esc_Connected - Connected!.
20:14.08.00-693: T= 1 : ServiceManager - Started in culture Swedish
(Sweden) with codepage 1252
20:14.08.00-693: T= 1 : ServiceManager - Starting thread for maintenance of
subscriptions...
20:14.08.00-693: T= 1 : ServiceManager - Starting thread for maintenance of
data access clients...
20:14.08.00-693: T= 1 : SendAttach: ATTACH response is:
AQR01;fc=ATTACH;rc=0;supports=checkcall,wildbookings,tempoptions,dutycodes;i
d=0;qptime=1497960479;jobzok=1;qpver= Trio Agent QP TE 7.1.335.18 2017-06-19
10:58 (QP 7.1.18 build 69);auservice=http://swe-ccapi-
test.edge.local:34000/core;amon=1;amonid=1;astamp=0;ccid=1
20:14.08.00-693: T= 1 : QPConnManager: QP attached
...

```

5.1.3 CD API backend

The CD API backend process *DirectoryAPI.Service.Windows.exe* logs to the file
X:\TE\ProgramData\CD1\log\DirectoryAPI.Service.Windows.log.

Example of a typical start-up log:

```

23:09.22.24-577: T= 1 : DirectoryAPI.Service.Windows version 0.0.705.1
running.
23:09.22.24-593: T= 1 : Connection String for DBAccessor:
Datasource=swerd2016vmwtpl.edge.local;Database=PresDB;Ini=C:\TE\ProgramFiles
\0.0.9007\bin\CD\Server\sql.ini;User
ID=server;Password=server;ClientName=DirectoryAPI.Service.Windows;isolation=
RL;
23:09.22.24-593: T= 1 : Directory SQLWNTM.DLL:
C:\TE\ProgramFiles\0.0.9007\bin\CD\Server
23:09.22.24-593: T= 1 : <K> SQLBaseConnection:
Datasource=swerd2016vmwtpl.edge.local;Database=PresDB;Ini=C:\TE\ProgramFiles
\0.0.9007\bin\CD\Server\sql.ini;User
ID=server;Password=server;ClientName=DirectoryAPI.Service.Windows;isolation=
RL;
23:09.22.24-609: T= 1 : Opening SQLBase Connection
23:09.22.24-671: T= 1 : SQLBase Connection opened
23:09.22.24-749: T= 1 : DBAccessor::Update UPDATE SYS_SVRPROCESSES SET
PID=11844 WHERE @UPPER(PROCESS)=@UPPER('DirectoryAPI.Service.Windows.exe')
AND @UPPER(NODEHOST)=@UPPER('SWERD2016VMwtpl')

```



```

23:09.22.24-749: T= 1 : Starting EventBroker Connection to 127.0.0.1:8899
23:09.22.24-874: T= 1 : GetParameterValue: (MissingDef)
CoreSecurity.IdentityProvider.Address
23:09.22.29-765: T= 6 : Connection String for DBAccessor:
Datasource=swerd2016vmwtp1.edge.local;Database=PresDB;Ini=C:\TE\ProgramFiles
\0.0.9007\bin\CD\Server\sql.ini;User
ID=server;Password=server;ClientName=Subscriber;isolation=RL;
23:09.22.29-765: T= 6 : Directory SQLWNTM.DLL:
C:\TE\ProgramFiles\0.0.9007\bin\CD\Server
23:09.22.29-765: T= 6 : <K> SQLBaseConnection:
Datasource=swerd2016vmwtp1.edge.local;Database=PresDB;Ini=C:\TE\ProgramFiles
\0.0.9007\bin\CD\Server\sql.ini;User
ID=server;Password=server;ClientName=Subscriber;isolation=RL;
23:09.22.29-765: T= 6 : Opening SQLBase Connection
23:09.22.29-765: T= 6 : SQLBase Connection opened
23:09.22.29-765: T= 6 : Creating object of type:
Trio.CompanyDirectory.APILibrary.MyEventBrokerConnection
23:09.22.29-781: T= 6 : Starting EventBroker Connection to
swerd2016vmwtp1:8899
23:09.22.30-843: T= 6 : Read TDL configuration. Initiating TDL with this
configuration
OAM=OAMA;PresDB=Present;Storage=EntityFramework.SqlServer|TdlSpecifiedLogin:
True,TdlUserId:trio,TdlPassword:trio,TdlServerId:swerd2016vmwtp1\EISQL,TdlDa
tabaseName:TrioDataLayer;AgentDP=CCDP;AgentQP=AgentQP;
23:09.22.32-656: T= 6 : Config changed: -1->5

```

Example of a licence problem log:

```

29:16.44.29-731: T= 1 : *** Error *** Exit due to ConfigProblem (No CD API
license)

```

5.1.4 Additional log files

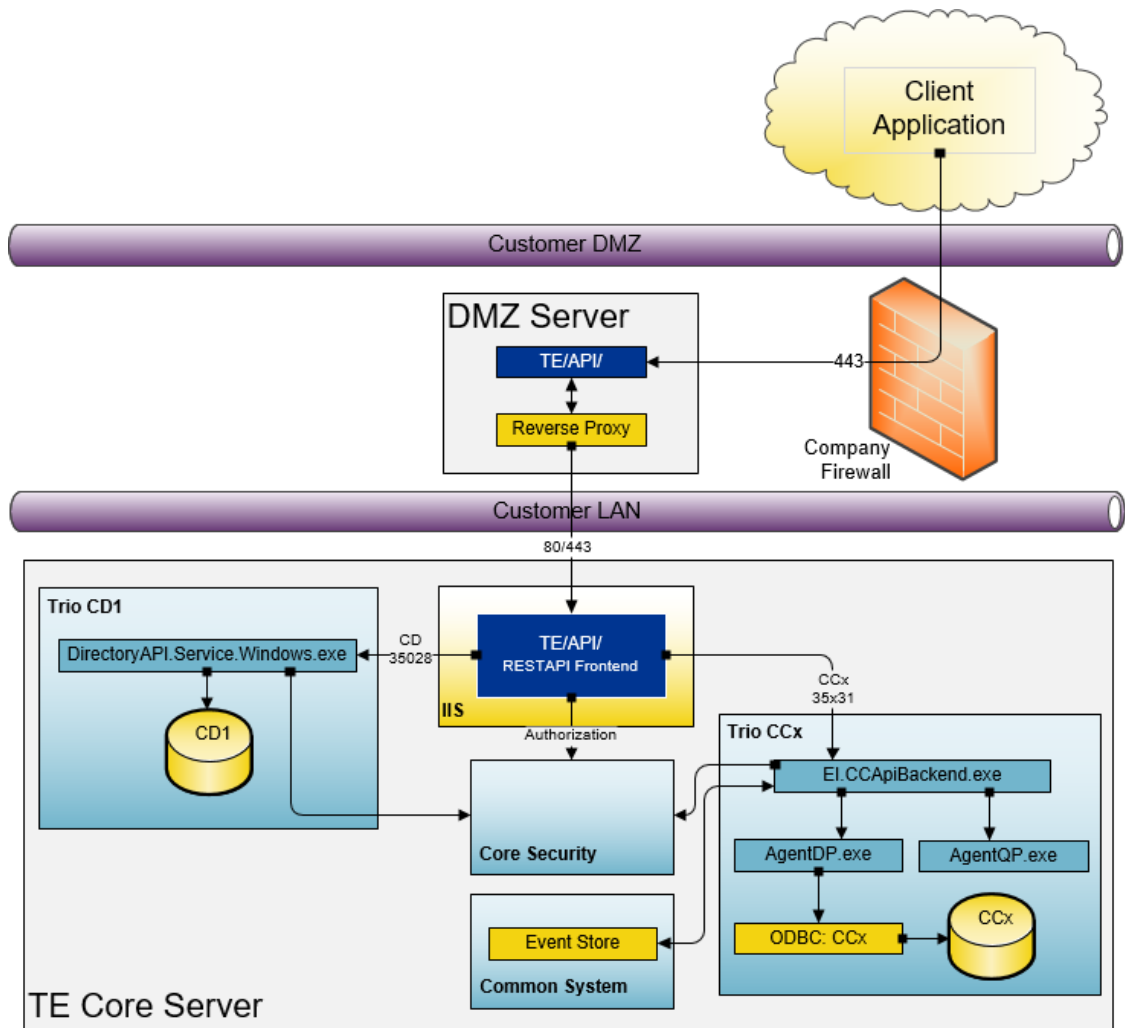
In addition to the logs for the REST API frontend and backend, there are some third party log files that could be of interest when troubleshooting.

| Module | Log file |
|------------|--|
| EventStore | X:\TE\ProgramData\CommonSystem\log\YYY-MM-DD\127.0.0.1-2113-cluster-node.log |

Appendix A – DMZ server

Introduction

If the TE API should be accessible externally, it is recommended that it is done through a reverse proxy in the DMZ. This appendix describes the rules that need to be configured to enable external access to the API through a reverse proxy. In addition, it contains an example of how secure communication and the reverse proxy can be configured using IIS.



If Trio Administrator Web has been installed on a DMZ server, that server may also be used to access the TE API, without any further configuration.

Proxy server rules

All URIs used when communicating with the TE API without a reverse proxy begin with `<te_core_server>/te/api` and the relative links that are returned from the API all starts with `/te/api/`.

To configure the reverse proxy to relay these requests to the TE core server, an application called TE/API should be added.

Uri used by the application when calling the TE server via the proxy

```
http://<proxy>/te/api/ or https://<proxy>/te/api/
```

Endpoint in the TE system

```
http://<teserver>/te/api/ or https://<teserver>/te/api/
```

Conclusion

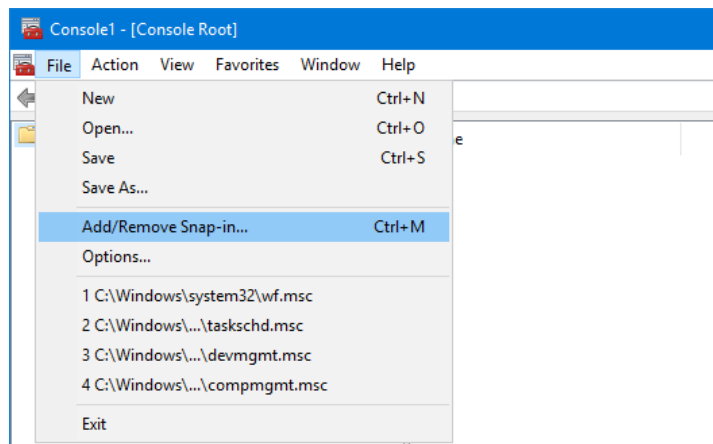
```
http://<proxy>/te/api/ or https://<proxy>/te/api/  
forward all requests to  
http://<teserver>/te/api/ or https://<teserver>/te/api/
```

Example – Configuration using IIS

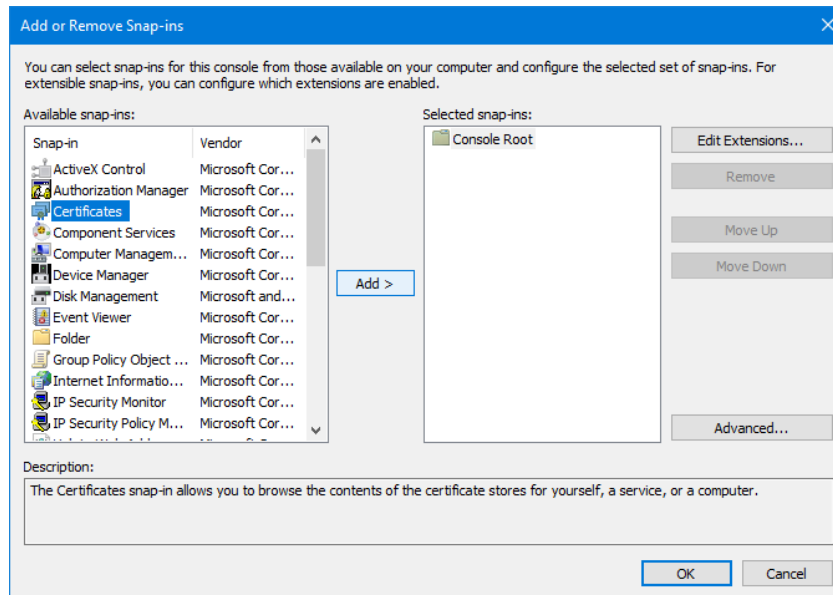
Secure communication

Import the certificate

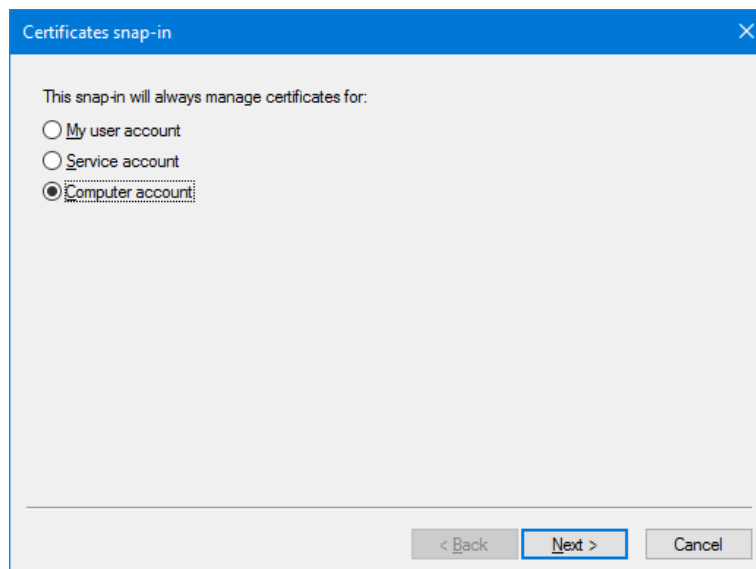
1. Place the customer's certificates on the server.
2. Start **MMC** and select **File -> Add/Remove Snap-in**.



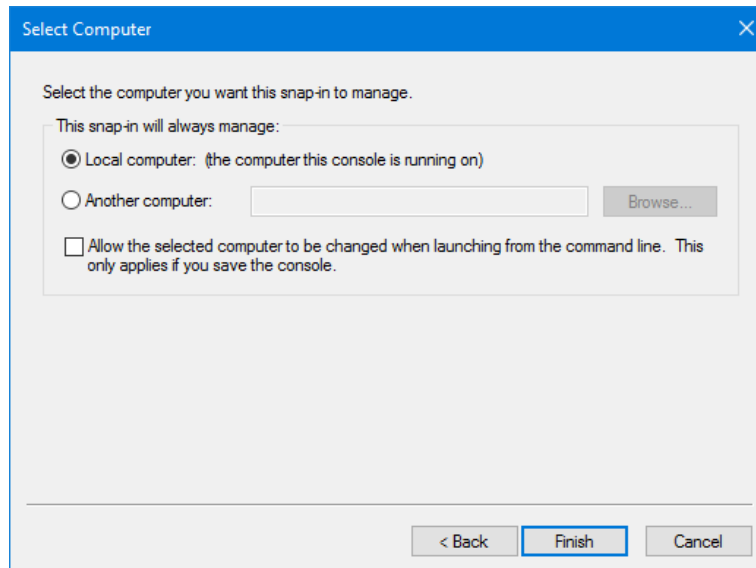
3. Select **Certificates** in the list to the left and click **Add**.



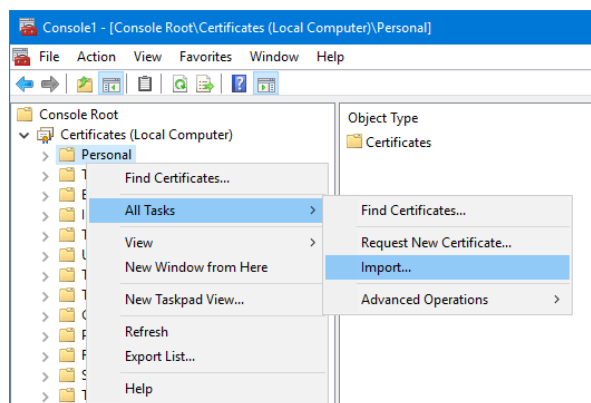
4. Select **Computer account** and click **Next**.



5. Select **Local computer** and click **Finish**.



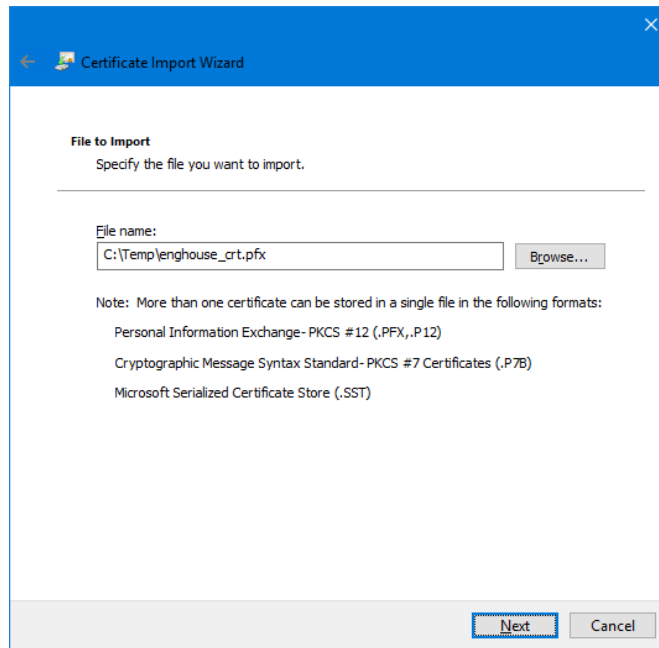
6. Click **OK** to exit the **Add or Remove Snap-ins** dialog.
7. Expand the tree structure, right click **Personal** and select **All Tasks -> Import....**



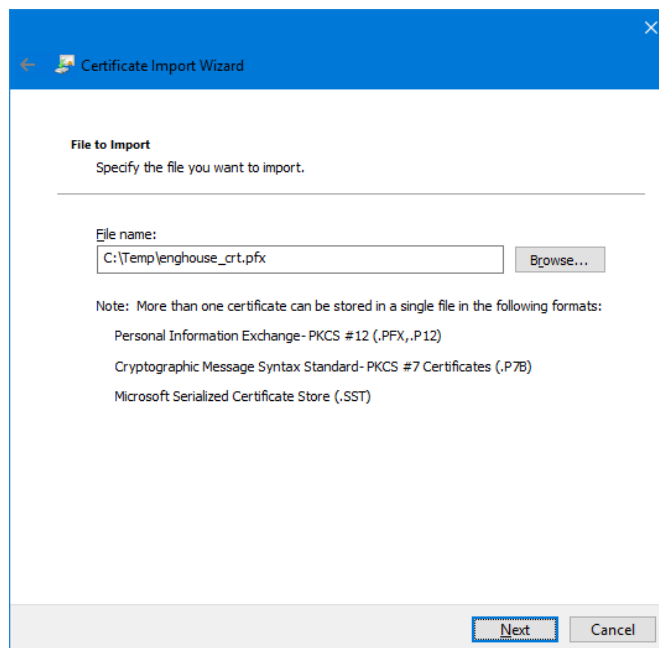
The **Certificate Import Wizard** starts.

8. Click **Next**.

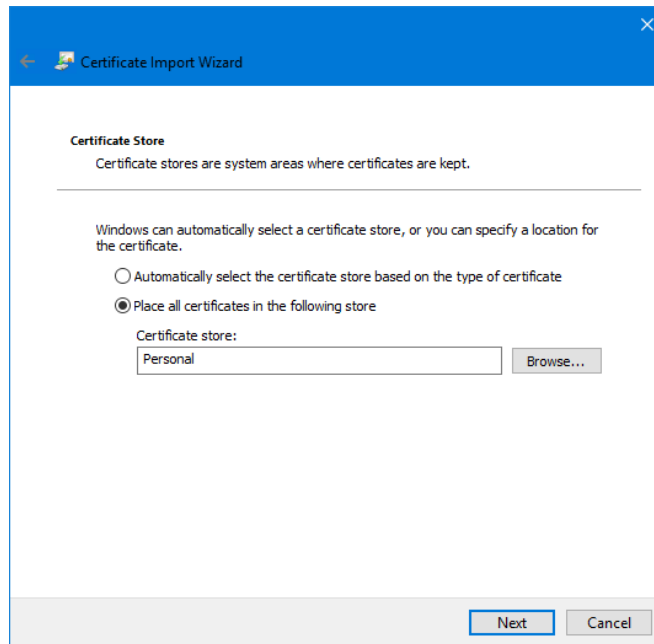
9. Browse to the certificate file and click **Next** again.



10. Enter the certificate's password and click **Next**.



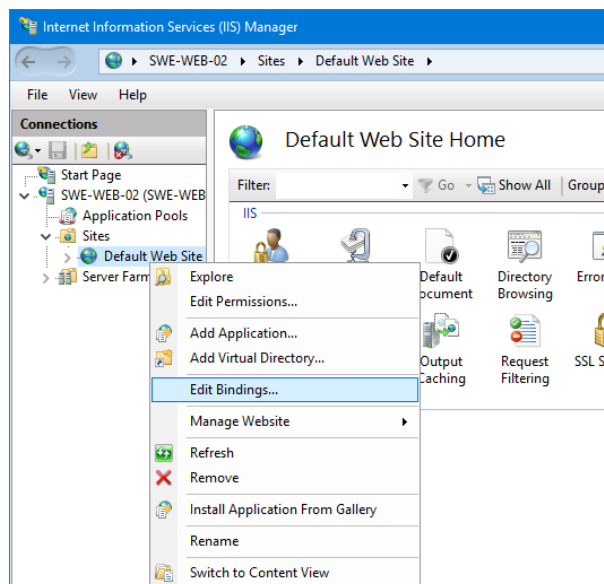
11. Accept that the certificate is placed in the **Personal** certificate store and click **Next**.



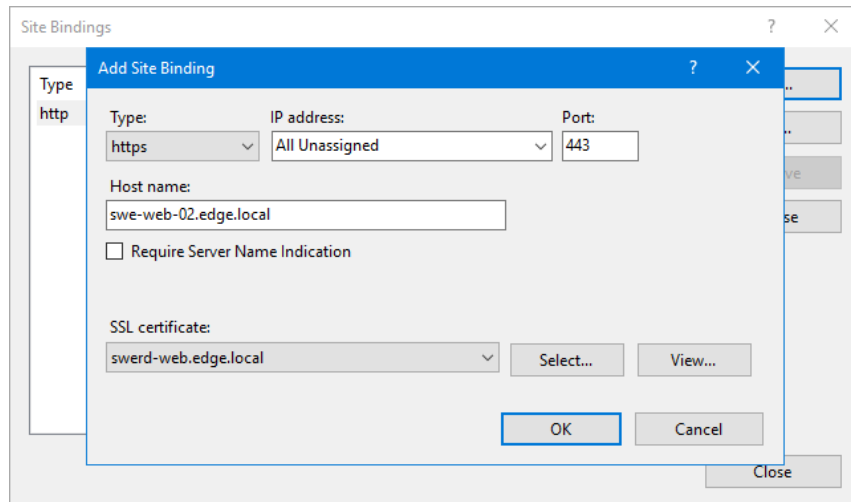
12. Click **Finish**.

Bind the certificate

1. Start **Internet Information Services (IIS) Manager**.
1. Right click **Default Web Site** and select **Edit Bindings**.



- Click **Add...** and enter settings according to the table below. Click **OK** when done.



| Parameter | Description |
|-----------------|---|
| Type | https |
| IP address | All Unassigned or a specific address. |
| Port | 443 |
| Host name | The public domain that should be used by the client application. If the certificate is for a specific domain name only (not *), the hostname is the same as the SSL certificate. |
| SSL certificate | Select the certificate that should be used. |

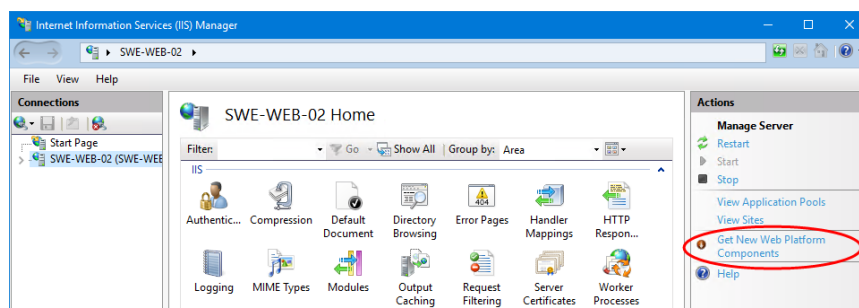
- Click **Close**.

Reverse proxy using IIS

Install ARR 3.0

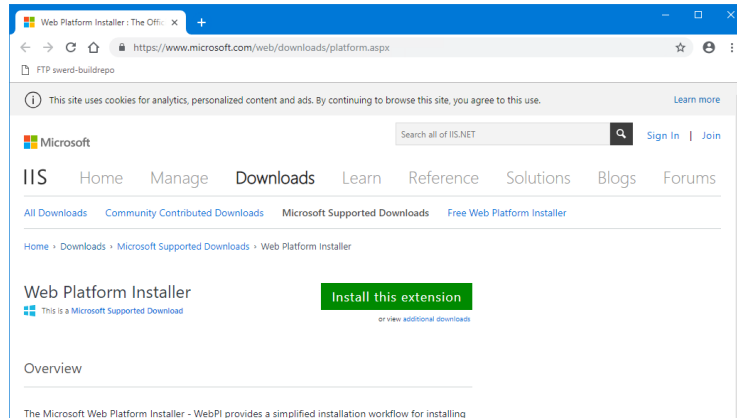
To be able to configure the reverse proxy in IIS, ARR 3.0 must first be installed.

- Start **Internet Information Services (IIS) Manager**.
- Highlight the server in the left pane and click **Get New Web Platform Components** in the right pane.

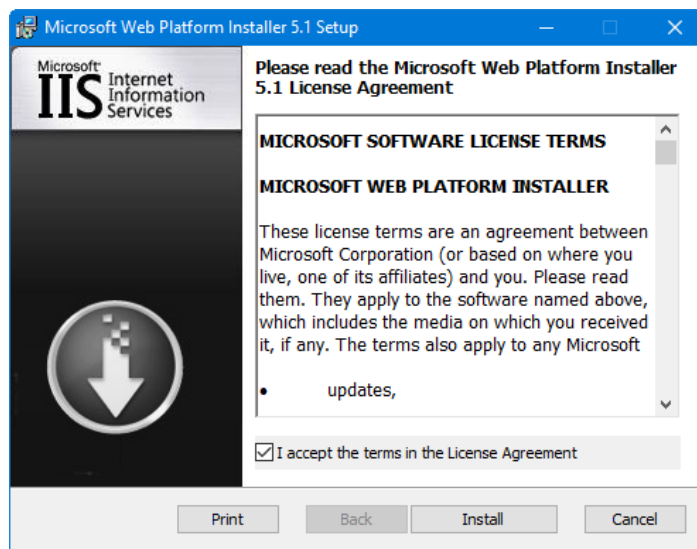


A website opens in the web browser.

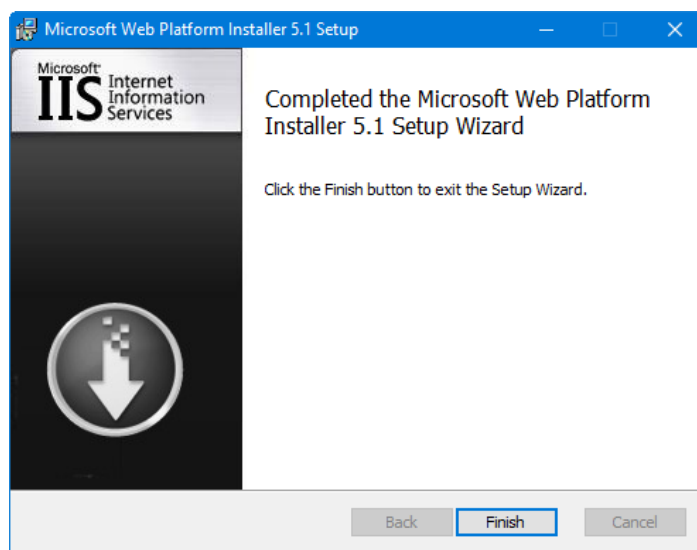
3. Click **Install this extension** and run the downloaded file **WebPlatformInstaller**.



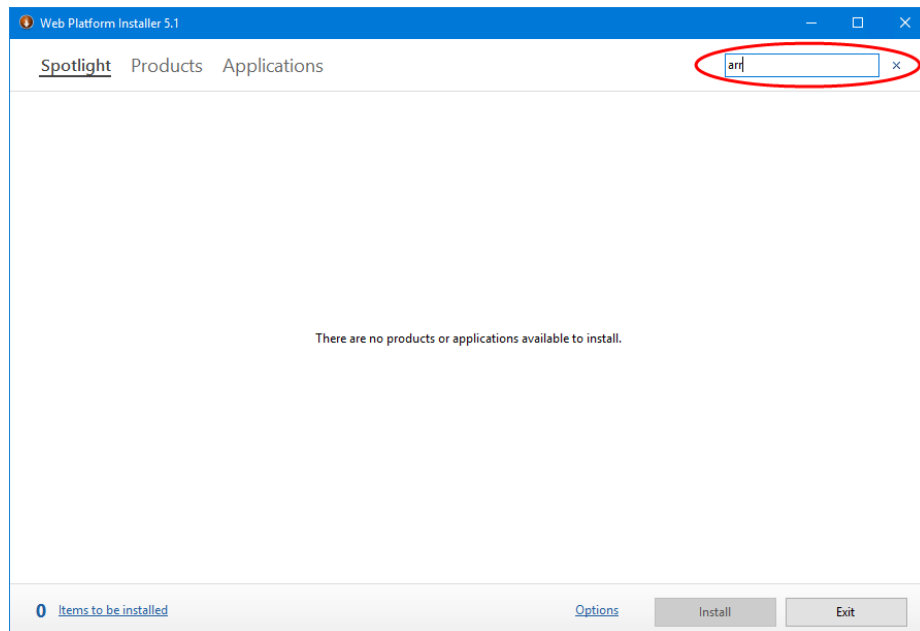
4. Accept the license agreement and click **Install**.



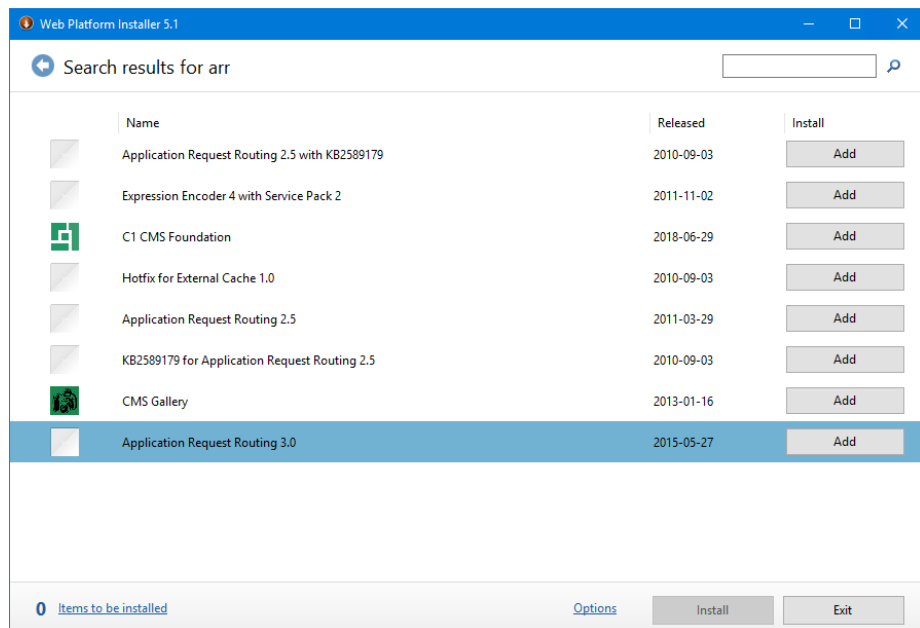
5. Click **Finish**.



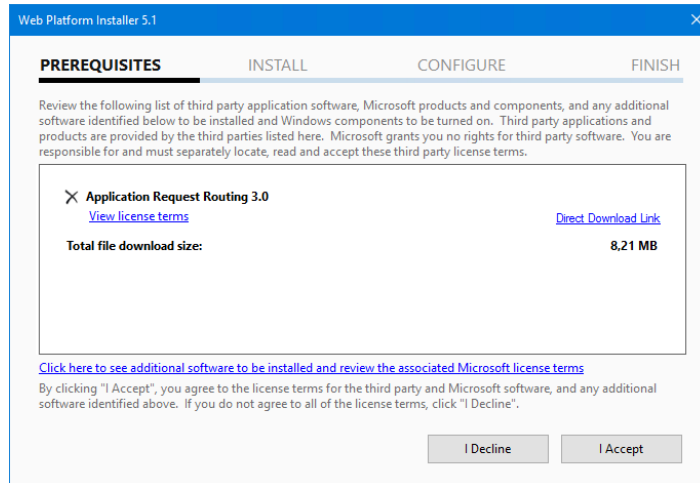
6. Return to **Internet Information Services (IIS) Manager** and click **Get New Web Platform Components** again to start the **Web Platform Installer**.
7. Enter **arr** in the search box in the upper right corner and press **Enter**.



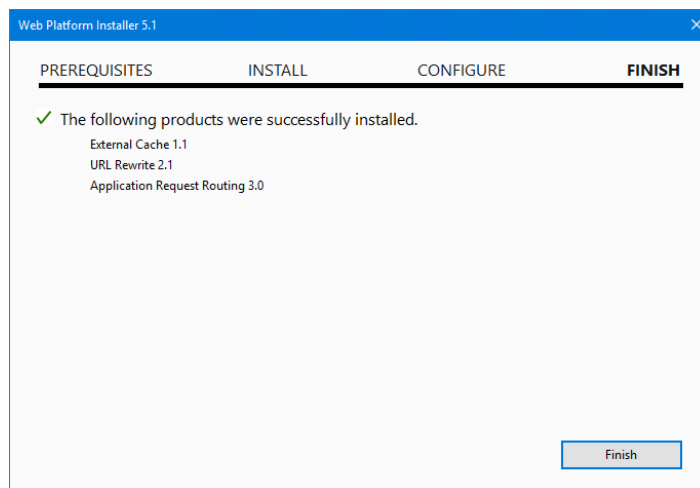
8. Find **Application Request Routing 3.0** and click **Add**. Then, click **Install**.



9. Click **I Accept** and wait for the installation to complete.

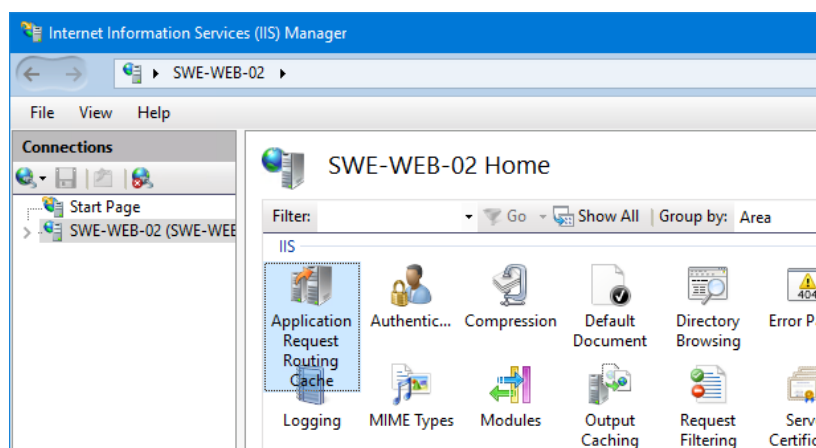


10. Click **Finish**, exit **Web Platform Installer** and close **Internet Information Services (IIS) Manager**.

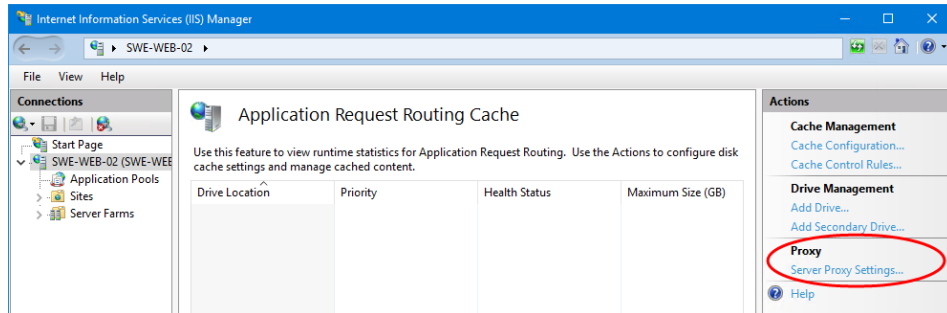


Enable proxy

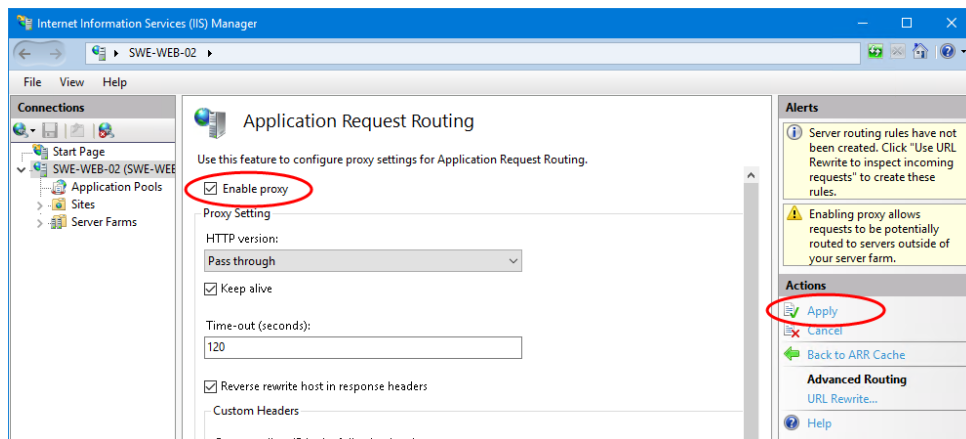
1. Start **Internet Information Services (IIS) Manager**.
2. Highlight the server in the left pane and double-click on **Application Request Routing Cache**.



3. Click **Server Proxy Settings** in the right pane.

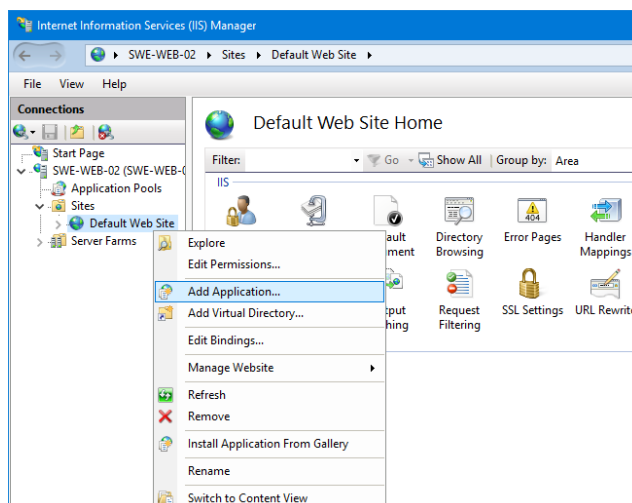


4. Check **Enable Proxy** and click **Apply** in the right pane.

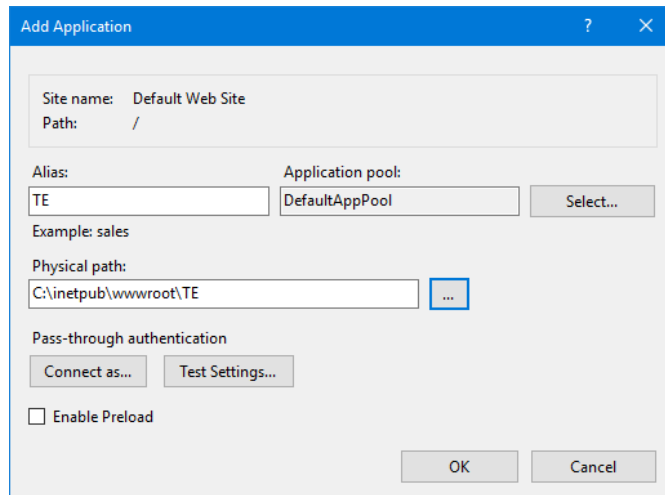


Add applications

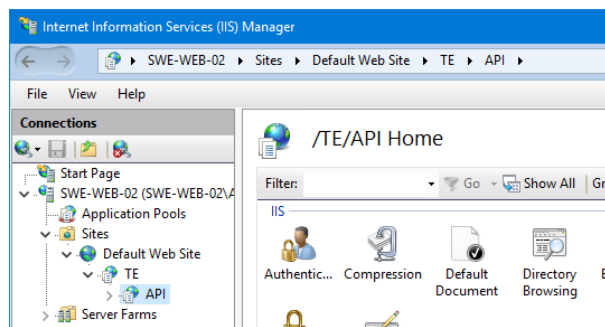
1. Expand the tree, right click **Default Web Site** and select **Add Application....**



2. Enter **TE** as **Alias** and create a new empty folder for the **Physical path**.



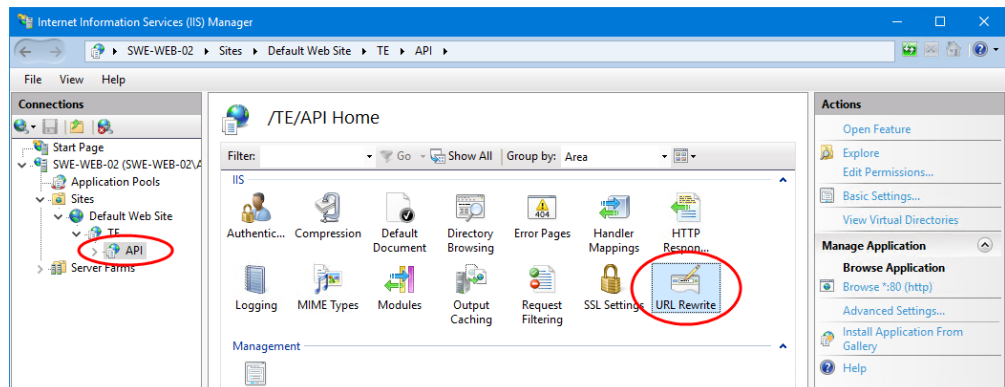
3. Click **OK** when done.
4. Right click the new application **TE** and select **Add Application....**
5. Enter **API** as **Alias** and create a new empty folder for the **Physical path**.
6. Click **OK** when done.



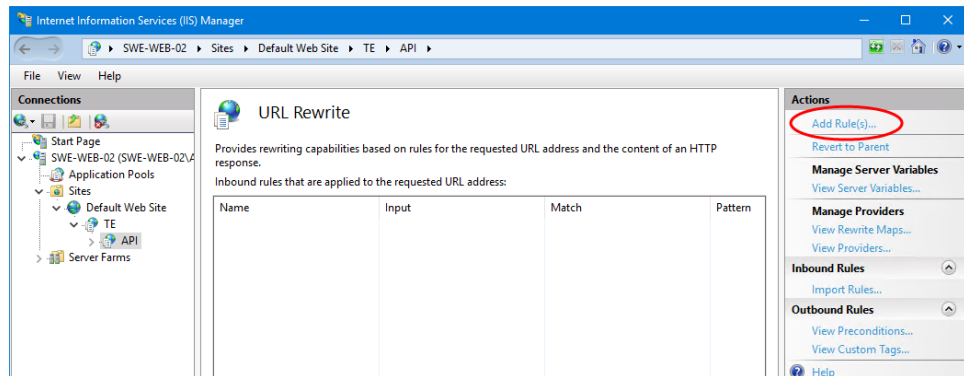
Note: If the new applications are not displayed in the tree to the left, press **F5** to refresh the view.

Configure URL rewrite

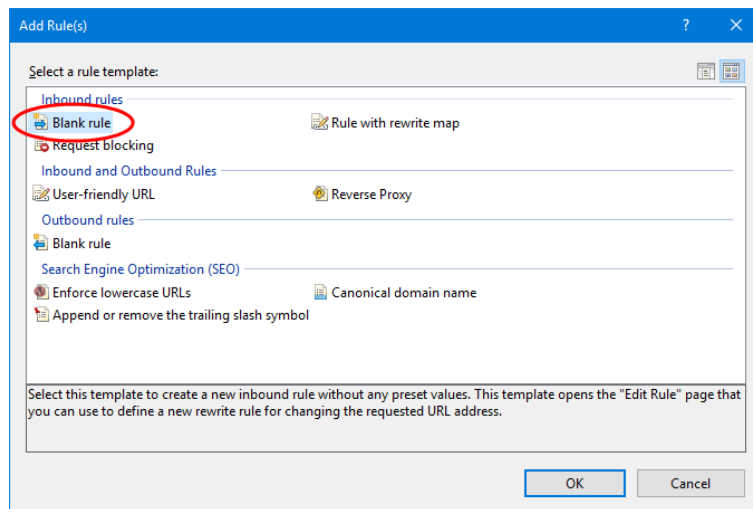
1. Highlight the application **TE/API** in the left pane and double click **URL Rewrite**.



- Click **Add rule(s)** in the right pane.



- Under **Inbound rules**, select **Blank rule** and click **OK**.



- Enter settings according to the table below. Parameters that are not mentioned in this table should not be altered from their default settings.

| Parameter | Value |
|-------------------------------------|--|
| Name | Give the rule a descriptive name. |
| Pattern | (.*) |
| Rewrite URL | http:// <te_core_server> /te/api/{R:1} Note: Replace <te_core_server> with the FQDN or IP address of the TE core server. |
| Stop processing of subsequent rules | Enabled |

Edit Inbound Rule

Name: ForwardToTrio

Match URL

Requested URL: Matches the Pattern Using: Regular Expressions

Pattern: (.*) [Test pattern...](#)

☒ Ignore case

Conditions

Server Variables

Action

Action type: Rewrite

Action Properties

Rewrite URL: http://sword-al-te1/te/api/{R:1}

☒ Append query string

☐ Log rewritten URL

☒ Stop processing of subsequent rules

Features View Content View

5. Click **Apply** when done.

Appendix B – Field identities

This appendix contains lists of the fields that can be used against the API, along with descriptions and type information. CC fields are not described in this appendix.

Subscriber data

A list of existing subscribers in the directory can be fetched from `/TE/API/directory/subscribers` (get). To manage a single subscriber, the path is `/TE/API/directory/subscribers/{subscriberId}` (get, post, patch).

The available fields in both cases are listed below:

| Field name (ID) | TE API field name | Permitted usage | Type | Chars | Description |
|-----------------------------------|-------------------|------------------|--------|-------|--|
| User id (101) | SubscriberId | get | Int | | The subscriber's unique id. |
| Cardkey number (102) | CardKey | get, post, patch | String | 30 | The subscriber's cardkey. |
| Christian name (103) | FirstName | get, post, patch | String | 30 | The subscriber's first name. |
| Surname (104) | LastName | get, post, patch | String | 30 | The subscriber's last name. |
| Title (105) | Title | get, post, patch | String | 100 | The subscriber's title. |
| Customer group (109) | CustGroup | get, post, patch | String | 20 | The customer group the subscriber belongs to. |
| Lock client customer group (1213) | CustGroupLock | get, post, patch | Bool | | True/false Is the subscriber locked to its own customer group? |
| Department no path | Department | get | String | 250 | The department that the subscriber belongs to. |
| Department (110) | DepartmentPath | post, patch | String | 250 | Complete department path. Department levels are entered separated with comma. Each level can be 250 chars. The departments are created if they do not already exist. |
| DepartmentId | DepartmentId | get, post, patch | Int | | Unique id of the department that the subscriber belongs to. |
| Extensions | Extensions | get, post, patch | Array | | A collection of the subscriber's extensions. See Extensions for details. |
| Location (112) | Location | get, post, patch | String | 50 | The subscriber's location/room. |
| Mobile phone (118) | Mobile | get, post, patch | String | 30 | The subscriber's mobile phone. |
| SMTP mail address (816) | Email | get, post, patch | String | 254 | The subscriber's email address. E.g. john.doe@trio.se |

| Field name (ID) | TE API field name | Permitted usage | Type | Chars | Description |
|------------------------------------|--------------------|------------------|--------|-------|--|
| SMS address (817) | SMS | get, post, patch | String | 254 | The subscriber's phone number for SMS. E.g. +46701234567 |
| Signature (1209) | Signature | get, post, patch | String | 30 | Signature for entering referrals. |
| Picture file name (1103) | PictureFile | get | String | 128 | The name of the subscriber's picture file (without path). |
| Subject (702) | Skills | get, post, patch | Array | 248 | A collection of the subscriber's skills. Each skill is a string with max. 248 chars. |
| User name (1203) | User | get, post, patch | String | 30 | The subscriber's Trio Enterprise username. Note: This field must always be combined with Password |
| Password (1204) | Password | post, patch | String | 30 | The subscriber's password Note: This field must always be combined with User name |
| Domain User Name (1218) | DomainUser | get, post, patch | String | 64 | The subscriber's windows domain name, e.g. TRIO\jdoe |
| Communicator Sign-in address (156) | presenceUri | get, post, patch | String | 60 | URI for presence e.g. from Cisco, Skype for Business or Teams. E.g. sip:john.doe@company.se |
| Mobile line state URI (163) | mobileLineStateUri | get, post, patch | String | 60 | URI for mobile line state, e.g. sip:john.doe@company.se |
| CalendarSystemId (2304) | calendarId | get, post, patch | Int | | ID of the calendar system the user belongs to. See Calendars . |
| CalendarUserName (2302) | calendarUserName | get, post, patch | String | 64 | Username in the calendar system, e.g. john.doe@company.se or John Doe/SE/TRIO. |
| Properties | Properties | get, post, patch | Array | | A collection of the subscriber's capabilities. Each capability consists of a field name (string) and its value (boolean). Post and patch can only be used for the agent field. See Properties for details. |
| Access profile Id (1216) | AccessProfileId | get | Int | | The ID of the subscriber's permission profile. |
| Profile Category Id | profileCategoryId | get | Int | | The ID of the category for the Access profile Id. |
| CC Id (1217) | CCId | get, post, patch | Int | | Which contact center the subscriber belongs to (only applicable for agents): -1 = All 0 = None 1-n = The specified CC |
| References (6601, 6602) | References | get | Array | | A collection of the subscriber's external references. Each reference consists of two keys: SystemId and ExternalId. To set the values, use the fields externalId and systemId. |

| Field name (ID) | TE API field name | Permitted usage | Type | Chars | Description |
|---------------------------|-------------------|------------------|--------|-------|---|
| External User Id (6602) | ExternalId | post, patch | String | 100 | The subscriber's identification in the external system. Must be specified together with External System Id. To get the value, use the References field. |
| External System Id (6601) | SystemId | post, patch | String | 4 | Id of the external system that the subscriber's external user ID belongs to. Must be specified together with External User Id. To get the value, use the References field. |
| ReferralCode | ReferralCode | get | String | 3 | Code for the subscriber's currently active referral. |
| ReferralStatus | ReferralStatus | get | String | 1 | The subscriber's referral status: 0=No active referral 1=Extension open 2=Do not disturb |
| ReferralToTime | ReferralToTime | get | String | 21 | When the last consecutive referral ends (i.e. the next time the subscriber will be available) Format: YYYY-mm-dd HH:MM:DD 'Until further notice' represents by the value '1901-01-01 00:59:XX'. |
| Extra field 1 (302) | Extra1 | get, post, patch | String | 254 | Customer specific information |
| Extra field 2 (303) | Extra2 | get, post, patch | String | 254 | Customer specific information |
| Extra field 3 (304) | Extra3 | get, post, patch | String | 254 | Customer specific information |
| Extra field 4 (305) | Extra4 | get, post, patch | String | 254 | Customer specific information |
| Extra field 5 (306) | Extra5 | get, post, patch | String | 254 | Customer specific information |
| Extra field 6 (307) | Extra6 | get, post, patch | String | 254 | Customer specific information |
| Extra field 7 (308) | Extra7 | get, post, patch | String | 254 | Customer specific information |
| Extra field 8 (309) | Extra8 | get, post, patch | String | 254 | Customer specific information |
| Extra field 9 (310) | Extra9 | get, post, patch | String | 254 | Customer specific information |
| Extra field 10 (311) | Extra10 | get, post, patch | String | 254 | Customer specific information |
| Extra field 11 (312) | Extra11 | get, post, patch | String | 254 | Customer specific information |
| Extra field 12 (313) | Extra12 | get, post, patch | String | 254 | Customer specific information |
| Extra field 13 (314) | Extra13 | get, post, patch | String | 254 | Customer specific information |
| Extra field 14 (315) | Extra14 | get, post, patch | String | 254 | Customer specific information |
| Extra field 15 (316) | Extra15 | get, post, patch | String | 254 | Customer specific information |
| Extra field 16 (317) | Extra16 | get, post, patch | String | 254 | Customer specific information |
| Extra field 17 (318) | Extra17 | get, post, patch | String | 254 | Customer specific information |
| Extra field 18 (319) | Extra18 | get, post, patch | String | 254 | Customer specific information |
| Extra field 19 (320) | Extra19 | get, post, patch | String | 254 | Customer specific information |
| Extra field 20 (321) | Extra20 | get, post, patch | String | 254 | Customer specific information |

| Field name (ID) | TE API field name | Permitted usage | Type | Chars | Description |
|-----------------|-------------------|------------------|--------------|-------|--|
| ChangerName | ChangerName | get | string | | The name of the person who did the last change on this subscriber. |
| ChangeTime | ChangeName | get | DateT ime | | The DateTime when the last change on this subscriber was performed. |
| HideUser | HideUser | get, post, patch | int | | 0=Subscriber not hidden 1=Subscriber hidden in user web, guest web and Auto Attendant |

Properties

The fields below can be handled in the `Properties` collection of the `subscriber` resource.

| API field name | Permitted usage | Type | Description |
|----------------|------------------|------|---|
| Agent | get, post, patch | Bool | True/false Is the subscriber an agent in the Contact Center? |
| User | get | Bool | True/false Is the subscriber a user with a username/password? Implicitly set to true when adding a user/password to the subscriber (unless the subscriber is an agent). |
| Extension | get | Bool | True/false Is the subscriber only an extension? |
| API | get | Bool | True/false Is the subscriber an API account? |
| NameSearch | get, post, patch | Bool | True/false Is the subscriber searchable by name? |

Extensions

The fields below are present in the `extensions` field of the `subscriber` resource, but a list of extensions for a subscriber can also be fetched separately through the `extensions` resource, at `/TE/API/directory/subscribers/{subscriberId}/extensions` (get) and for managing a single extension at `/TE/API/directory/subscribers/{subscriberId}/extensions/{extensionId}` (get, post, patch).

| Field name | TE API field name | Permitted usage | Type | Chars | Description |
|--------------------|-------------------|------------------|--------|-------|--|
| ExtensionId | ExtensionId | get | Int | | Unique id of the extension. |
| Extension number | Nr | get, post, patch | String | 20 | The extension number. |
| PBX number | PbxId | get, post, patch | Int | | The ID of the PBX the extension belongs to. |
| Extension priority | Priority | get, post, patch | Int | | 1-n, 1=owner of the extensions, 2-n = associated with the extension. Default=1 |

Referrals

The subscriber's currently active referral is available in the `subscriber` resource, but a list of all referrals for a subscriber can also be fetched separately through the `referrals` resource, at `/TE/API/directory/subscribers/{subscriberId}/referrals` (get) and for managing a single referral at `/TE/API/directory/subscribers/{subscriberId}/referrals/{referralId}` (get, post, patch).

| Field name | TE API field name | Permitted usage | Type | Chars | Description |
|---------------------|--------------------|------------------|----------|-------|---|
| Referral ID | Id | get | Int | | Unique id of the referral. |
| From time | From | get, post, patch | DateTime | | The time when the referral is to be activated. If left empty in post/patch, 'now' is used. |
| To time | To | get, post, patch | DateTime | | The time when the referral will be deactivated. If left empty in post/patch, the referral is active until further notice. |
| Code | Code | get, post, patch | Int | | Unique id of the referral code. |
| Referral code | ReferralCode | get | String | 3 | Short text identifier of the code. |
| Referral text | ReferralText | get | String | 20 | Text description of the code. |
| Status | Status | get, post, patch | Int | | 0 = notActive 1 = openActive 2 = closedActive |
| Refer type | ReferType | get, post, patch | Int | | 1 = Closing 2 = NonClosing 3 = ClosingManualOpen |
| Forward type | ForwardType | get, post, patch | Int | | 0 = Unknown 1 = Normal 2 = ToVoicemail 3 = ToMobile 4 = ToAlternateNumber Note: Type 4 requires a valid alternate answering destination |
| Alternate answering | AlternateAnswering | get, post, patch | String | 30 | If forward type is 4 (ToAlternateNumber), this field must contain the alternate phone number. |
| Information | Information | get, post, patch | String | 254 | Optional information about the referral to be presented in the client applications. |
| Online status | OnlineStatus | get, post, patch | Int | | Not used in this version of Trio Enterprise. |
| Signature | Signature | get | String | 30 | The signature of the user who created/updated the referral. |
| Time stamp | TimeStamp | get | DateTime | | The time when the referral was created/updated. |

Organisation data

Departments

A list of existing departments in the directory can be fetched from the resource `/TE/API/directory/departments`. The following fields are available for each department:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|---|
| Id | get | | | Unique id of this department. |
| Name | get | String | 250 | Name of the department. |
| Level | get | Int | | 0-n, the level in the department tree where this department is located. Value of 0 means root department. |
| ParentId | get | Int | | 1-n, id of this departments parent in the department tree. -1 for root departments. |
| Idx | get | Int | | Order position in department tree. |

Note: Departments are set up in a tree. At the top level, there are one or multiple root departments. Each root department can have none or multiple sub departments, which in turn can have their own sub departments, and so on.

Skills

A list of existing skills in the directory can be fetched from the resource `/TE/API/directory/skills` (get). Each skill in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|-------------------------|
| Id | get | Int | | Unique id of the skill. |
| Name | get | String | 248 | Name of the skill. |

Titles

A list of existing titles in the directory can be fetched from the resource `/TE/API/directory/titles` (get). Each title in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|-------------------------|
| Id | get | Int | | Unique id of the title. |
| Name | get | String | 100 | Name of the title. |

Customer groups

A list of existing customer groups in the directory can be fetched from the resource `/TE/API/directory/customergroups` (get). Each customer group in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|----------------------------------|
| Id | get | Int | | Unique id of the customer group. |
| Name | get | String | 20 | Name of the customer group. |

Extra field names

A list of existing extra fields in the directory can be fetched from the resource `/TE/API/directory/extrafieldnames` (get). Each extra field in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|-------------------------------|
| Id | get | Int | | Unique id of the extra field. |
| Name | get | String | 64 | Name of the extra field. |

PBXs

A list of existing PBXs in the directory can be fetched from the resource `/TE/API/directory/pbxes` (get). Each PBX in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|--|
| Id | get | Int | | Unique id of the PBX. |
| Name | get | String | 16 | Name of the PBX. |
| SystemId | get | Int | | System id of the PBX, used to tie a subscriber with an <i>ExternalId</i> to a certain PBX. |

Calendar systems

A list of existing calendar integrations in the directory can be fetched from the resource `/TE/API/directory/calendarsystems` (get). Each calendar integration in the list has the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|--|
| Id | get | Int | | Unique id of the calendar integration. |
| Server | get | String | 64 | Internal name of the calendar integration. |
| Type | get | Int | | Type of calendar integration. |
| Name | get | String | 100 | Username for the calendar integration. |

Capabilities

A list of capabilities for the directory can be fetched from the resource

/TE/API/directory/capabilities (get). The following fields exist for this resource:

| TE API field name | Permitted usage | Type | Chars | Description |
|---------------------|-----------------|------|-------|---|
| presenceOrLinestate | get | Bool | | True/false Is presence and/or linestate integration enabled in the system? |
| guestWebEnabled | get | Bool | | True/false Is Guest Web enabled in the system? |

Security

Groups

A list of permission groups can be fetched using the resource

/TE/API/security/groups?subsystemId={ccid}&extId&resourceTypeId={id}

This resource supports filtering using the following fields:

| Query parameter | Type | Description |
|-----------------|------|--|
| SubsystemId | Int | Filters the returned data on the <i>SubsystemId</i> (ccid). |
| ExtId | Int | Filters the returned data on the <i>ExtId</i> . |
| ResourceTypeId | Int | Filters the returned data on the <i>ResourceTypeId</i> . Supported resource type IDs are: 1001 Agent 1120 SkillProfile. |

This example filters on agents in CC1:

/TE/API/security/groups?subsystemId=1&extId&resourceTypeId=1001

The following fields are returned in the result:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|-------|-------|--|
| Groups | get | Array | | A collection of permission groups. See Groups for details. |

Groups

The *Groups* array contains the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|--------|-------|---|
| Id | get | Int | | The resource ID of the group. |
| Name | get | String | 254 | The name of the group. |
| Members | get | Array | | A collection of the group members. See Members for details. |

Members

The *Members* array contains the following fields:

| Query parameters | Permitted usage | Type | Chars | Description |
|------------------|-----------------|--------|-------|---|
| Id | get | Int | | The resource ID of the member. |
| ExtId | get | Int | | The foreign key of the member. For an agent, this is the agent ID. For a skill profile, this is the skill profile ID. |
| Name | get | String | 254 | The name of the member. |
| SubsystemId | get | Int | | The id of the subsystem (ccId) where the member exist. |
| ResourceType | get | String | 254 | The name of the resource type. |
| ResourceTypeId | get | Int | | The ID of the resource type. |

Group members

The group members of a permission group can be updated using the REST resource `/TE/API/security/groups/{groupId}/members/{memberId}`

This resource can also be used to delete a group member:

`DELETE /TE/API/security/groups/{groupId}/members/{memberId}`

The following fields can be used in the POST request:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|------|-------|---|
| ExtId | post | Int | | The foreign key of the member. For an agent, this is the agent ID. For a skill profile, this is the skill profile ID. |
| ResourceTypeId | post | Int | | The ID of the resource type. |
| SubsystemId | post | Int | | The ID of the subsystem (ccid). |

Agents

The logged in user's effective permissions to agents can be fetched using the resource
`/TE/API/security/agents?subsystemId={ccid}&Agents={agid}`

The resource supports filtering on the following fields:

| Query parameters | Type | Description |
|------------------|--------|--|
| SubsystemId | Int | Filters the returned data on the <i>SubssytemId</i> (ccid). |
| Agents | String | Filters the returned data on these agents. Enter the agent IDs separated by comma. |

This example filters on the agents 1, 2 and 3 in CC1:

`/TE/API/security/agents?subsystemId=1&Agents=1,2,3`

The following fields are returned in the result:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|-------|-------|---|
| Subsystems | get | Array | | A collection of subsystems (CCs). See Subsystems below for details. |

Subsystems

The *Subsystem* array contains the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|-------|-------|---|
| Id | get | Int | | The ID of the subsystem (ccid). |
| ResourceTypeId | get | Int | | The ID of the resource type. In this case 1001 (agent). |
| Items | get | Array | | A collection of items. In this case, a collection of agents. See Items below for details. |

Items

The Items array contains the following fields, which describes the logged in user's permissions to the specific items:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|------|-------|---|
| Id | get | Int | | The item ID. In this case the agent ID. |
| CanBrowse | get | Bool | | Browse permission to this item (agent). |
| CanRead | get | Bool | | Read permission to this item (agent). |
| CanUpdate | get | Bool | | Update permission to this item (agent). |
| CanCreate | get | Bool | | Create permission to this item (agent). |
| CanDelete | get | Bool | | Delete permission to this item (agent). |

Services

The logged in user's effective permissions to services can be fetched using the resource `/TE/API/security/services?subsystemId={ccid}&services={sid}`

The resource supports filtering on the following fields:

| Query parameters | Type | Description |
|------------------|--------|--|
| SubsystemId | Int | Filters the returned data on the <i>SubssytemId</i> (ccid). |
| Services | String | Filters the returned data on these services. Enter the service IDs separated by comma. |

This example filters on the services 1, 2 and 3 in CC1:

`/TE/API/security/services?subsystemId=1&services=1,2,3`

The following fields are returned in the result:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|-------|-------|---|
| Subsystems | get | Array | | A collection of subsystems (CCs). See Subsystems below for details. |

Subsystems

The *Subsystem* array contains the following fields:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|-------|-------|---|
| Id | get | Int | | The ID of the subsystem (ccid). |
| ResourceTypeId | get | Int | | The ID of the resource type. In this case 1002 (service). |
| Items | get | Array | | A collection of items. In this case, a collection of services. See Items below for details. |

Items

The Items array contains the following fields, which describes the logged in user's permissions to the specific items:

| TE API field name | Permitted usage | Type | Chars | Description |
|-------------------|-----------------|------|-------|---|
| Id | get | Int | | The item ID. In this case the service ID. |
| CanBrowse | get | Bool | | Browse permission to this item (service). |
| CanRead | get | Bool | | Read permission to this item (service). |
| CanUpdate | get | Bool | | Update permission to this item (service). |
| CanCreate | get | Bool | | Create permission to this item (service). |
| CanDelete | get | Bool | | Delete permission to this item (service). |