TeamDrive Registration Server Reference Guide

Release 3.0.018.6

Eckhard Pruehs

February 24, 2015

1 Glossary **1**

2 Introduction **3**

3 Software components **5**

4 TeamDrive System Components **7**

5 Security **9**

6 Provider Concept **11**
6.1 The Provider Code . . . 11
6.2 The DISTRIBUTOR File for a Provider . . . 11
6.3 User Allocation . . . 11
6.4 Provider parameters . . . 12
6.5 Hosting Service for each Provider . . . 12
6.6 Client license keys . . . 12
6.7 API access . . . 13

7 TeamDrive Client <–> Server interaction **15**
7.1 User account . . . 15
7.2 Device . . . 20
7.3 Messages, Invitations & Invitation Types . . . 20
7.4 Emails . . . 22
7.5 Change User data . . . 22
7.6 Banner . . . 23
7.7 Updates . . . 23
7.8 Server URLs . . . 24
7.9 Initial Storage Account Request . . . 24

8 HTML and EMail Templates **25**
8.1 HTML Templates . . . 25
8.2 Email Templates . . . 26

9 TeamDrive Name Server (TDNS) **29**
9.1 Data security on the TDNS . . . 29
9.2 Communication workflow from Client to Registration Server to TDNS and the way back . . . 29

10 External Authentication **31**
10.1 External User Data . . . 31
10.2 Compelling Re-login . . . 32
10.3 Login Configuration . . . 32
10.4 Lost Password and Registration . . . 32
10.5 Authentication Examples . . . 33
10.6 Authentication Tokens and Verification Pages . . . 33
10.7 Login Procedure . . . 34

11 Databases **39**
11.1 Database “td2reg” . . . 39
11.2 Database “td2apilog” . . . 48

12 Auto Tasks **49**
12.1 “Send Emails”-Task . . . 49
12.2 “Delete Old Messages”-Task . . . 49
12.3 “Move Store Forward Messages”-Task . . . 49
12.4 “Delete Client IPs”-Task . . . 50
12.5 “Update RegServer-List”-Task . . . 50
12.6 “CleanUp”-Task . . . 50
12.7 “Expire Licenses”-Task . . . 50

13 Settings **51**
13.1 Registration Server Settings . . . 51
13.2 Provider Settings . . . 58
13.3 Login and Registration Client Settings . . . 67

14 Registration Server API **73**
14.1 API Basics . . . 73
14.2 Registration-Server API Calls . . . 76
14.3 Error Codes . . . 122

Index **125**

CHAPTER

ONE

GLOSSARY

Client The software application used by users to interact with the TeamDrive system. Can be customized to various degrees. Every device requires a client application.

Device A computer used by a user to access the TeamDrive system.

Installation Simply refers to the installation of the client application on a device.

User A person using the TeamDrive System.

Provider (aka Distributor or Tenant) The "owner" of some set of Users. See *Provider Concept* (page 11) for a detailed explanation.

Space A virtual folder containing data that can be shared with other TeamDrive users. This is what TeamDrive is all about.

CHAPTER

TWO

INTRODUCTION

The main functionality of the TeamDrive Registration Server is handling the public keys of the TeamDrive Clients and storing the invitations between users until they get downloaded by a TeamDrive Client.

Beside this functionality, the Registration Server also handles client licenses, sending emails for registration, activation and invitations, storing the default host server accounts for clients, and storing the individual Provider settings. The server can also update the banners in the free client and inform users about available client updates.

A Registration Server can also be a part of the TeamDrive Name Server (TDNS) Network which will allow clients to invite users which are registered at other Registration Server. The different parts will be described in the next chapters.

This documentation describes the functionality of the current release version 3.0.018 which supports external authentication. The chapters which belong to 3.0.018 will be marked in this document. You also need a recent client version to use it together with version 3.0.018 of the TeamDrive Registration Server.

CHAPTER

THREE

SOFTWARE COMPONENTS

The TeamDrive Registration Server is based on the following components:

- Linux 64 bit system
- MySQL Database Server 5.1 or newer
- Apache Web Server 2.2
- Apache registration server module & sources
- An open port 80 for incoming request and an open port 80 for outgoing request to connect Hosting Service and TDNS (if enabled)
- A PBAC background process which will handle recurring tasks, like sending mails (see *Auto Tasks* (page 49))

See the document "TeamDrive Registration Server Installation and Configuration" for detailed installation instructions.

CHAPTER

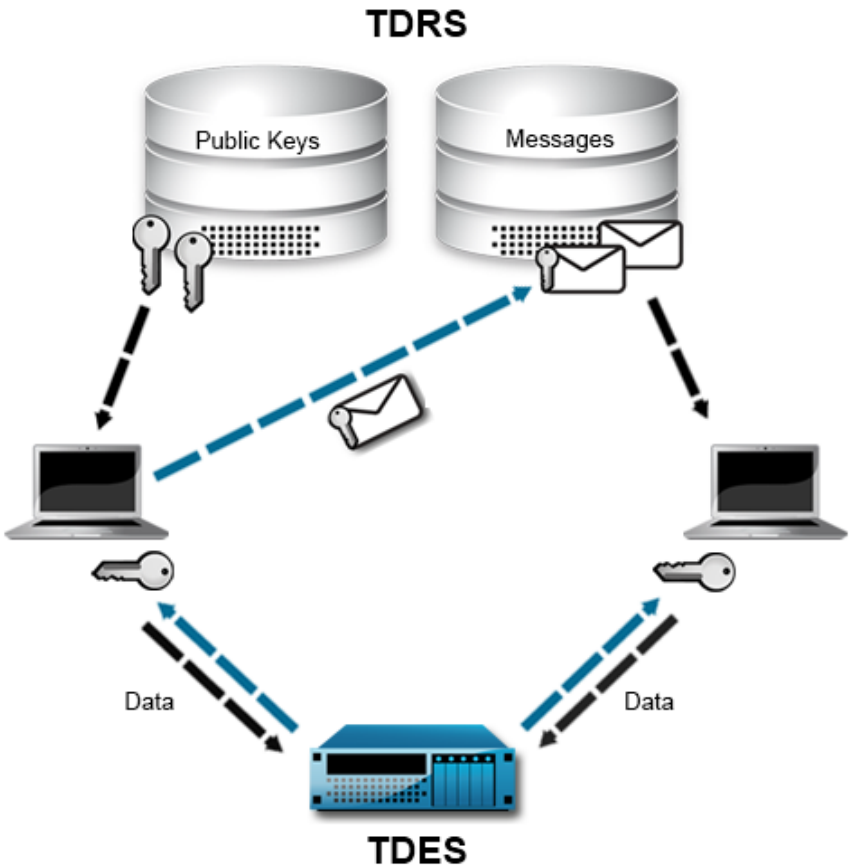
FOUR

TEAMDRIVE SYSTEM COMPONENTS

The TeamDrive Registration Server is the first component necessary to register a TeamDrive Client, but the Registration Server is only one part of the complete system. Several Registration Servers can be connected to the TDNS network which allows users from different Registration Servers to invite each other. For more details see *TeamDrive Name Server (TDNS)* (page 29).

The second important part is the TeamDrive Hosting Service. A TeamDrive Client can upload data from Space to a WebDAV Server, a TeamDrive Personal Server (TDPS), or an Enterprise Hosting Service (TDES). These are collectively known as Hosting Services.

The TeamDrive Enterprise Hosting Service is a scalable Hosting Service that manages storage and traffic of a large number of clients. This is not possible with the TDPS or a WebDAV Server. TDES also has an HTTP-based API which allows remote management.



In summary: a Hosting Service stores the data of a TeamDrive Client and a Registration Server will handle the invitations between different clients, so that the users can work together in their Spaces and share documents with each other. The Registration Server will never store documents of the users and the Hosting Service does not know how many or which clients are accessing the different Spaces.

CHAPTER

FIVE

SECURITY

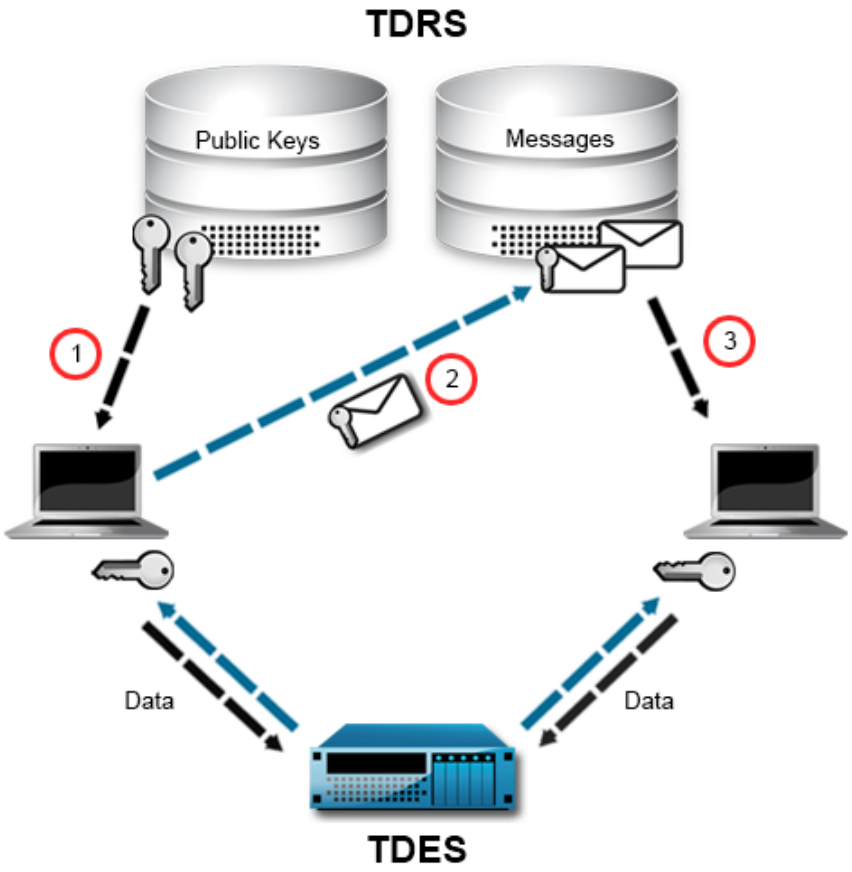
TeamDrive uses various technologies to make all communication secure. The TeamDrive Clients and the Registration Server use a Public-/Private Key mechanism to encrypt all data. The Registration Server generates a Public- and Private Key after the first start. A TeamDrive Client will ask for the Registration Server Public-Key when a user tries to register or login with a client. All requests from the client will be encrypted using a symmetric AES-256 key which will be generated based on the Public-Key of the Registration Server. The encrypted data can be send over standard HTTP without an additional SSL connection.

The Registration Server will decrypt the request using its Private Key. The type of answer returned to the client depends on the data returned. Most functions only return a success or failure informations, like checking a new license key. In this case the answer is not encrypted.

In other cases, like searching, a user will be encrypted. The way back to a client uses the Public-Key of the client installation. During the registration or login process, a TeamDrive Client will generate its own Public- and Private-Key. The Private-Key will be kept local in the local key store. The Public-Key will be uploaded to the Registration Server, so that the Public-Keys are available for other users.

Note: Users don't have personal Public-/Private-Key; every installation made by a user has its own Public-/Private-Key. This is important to understand, since even the different devices under the same user will have their own Public-/Private-Key.

A typical way how sharing data is working:



1. A user X will create a new Space on a WebDAV, TDPS or Hosting Service server. During the Space creation the client will create a new AES-256 Space key which will be used to encrypt all data in this Space
2. User X invites User Y to their Space. User X will generate an invitation document with the URL, access credentials, the AES-256 key of the Space, and other statistic informations for the Space. The user X's client looks for all registered devices of user Y on the Registration Server. The client checks if all Public-Keys of the devices are already in his local key store. If not, it will download the missing Public-Keys **(1)**. If all Public-Keys were not stored on the Registration Server, invitations could not be encrypted. Clients can only access the Registration Server. They cannot directly connect to each other.
3. User X encrypts the invitation document using the Public-Keys User Y's devices. Before the encrypted invitation is uploaded to the Registration Server, the request to the Registration Server is again encrypted using the Public-Key of the Registration Server **(2)**.
4. The Registration Server decrypts the request from the client and stores the encrypted invitation document in its database. It practically impossible to decrypt and read the encrypted invitation on the server side. The invitation document will be stored until User Y's client polls for new invitations.
5. The client of User Y downloads the invitation and decrypts the invitation document using their Private-Key **(3)**. Upon accepting the invitation, the data in the Space on the WebDAV, TDPS, or Hosting Service server will be downloaded and decrypted using the Space's AES-256 key (which was extracted from the invitation document).

CHAPTER

SIX

PROVIDER CONCEPT

A Provider is a partner or customer that "owns" a number of TeamDrive users. In turn, every TeamDrive user is associated with a particular Provider.

A Registration Server may have any number of Providers. Most Registration Server settings can be set per Provider. This means a Provider has significant control over its users. This includes the following:

- Client-side settings can be specified in order to configure login, registration, and to determine the behaviour of the client in general.
- Clicking links in the TeamDrive Client re-directed the user to Provider specific URLs.
- Users are directed to a Hosting Service or Registration Server that belongs to, or is associated with, the Provider.

6.1 The Provider Code

Each Provider has a globally unique Provider Code. The Provider Code is a 4 character sequence. The allowed characters are A to Z and 0 to 9. All new Provider Codes have to be approved by TeamDrive Systems GmbH.

The main TeamDrive Systems Provider Code is TMDR.

6.2 The DISTRIBUTOR File for a Provider

The `DISTRIBUTOR` file is part of the installation of a TeamDrive Client. The file is signed so that it cannot be altered after installation.

The `DISTRIBUTOR` file contains the Provider Code, a list of URLs that reference the Registration Server associated with the Provider, and a number of client settings.

On registration, the Provider Code in the `DISTRIBUTOR` file is sent the Registration Server. The code is then used in the process of "user allocation", as described below.

6.3 User Allocation

The Provider of a user is fixed at the moment they login or register. User allocation is generally determined by the Provider Code in the `DISTRIBUTOR` file or by the Provider Code panel in the first page of the client registration.

Providers with a TeamDrive OEM client should offer their own download site. These installations are packaged with their own `DISTRIBUTOR` file. This way, user's that download and install this version of TeamDrive are automatically allocated to that Provider.

6.3.1 Network Allocation

The process of Network Allocation can override user allocation determined by the `DISTRIBUTOR` file. In this case, the IP address of the TeamDrive Client is used to determine the Provider of the user.

Each Provider can specify its ownership of a number of IP networks (see `CLIENT_NETWORKS` setting in *CLIENT_NETWORKS (<ClientNetworks> tag)* (page 61)). If a TeamDrive Client is started in one of these networks the server can detect this from the IP address of the client and allocate the user to the Provider that owns the network. Network allocation has priority over `DISTRIBUTOR` file allocation.

In this way, it is not necessary for every Provider to have their own version of the TeamDrive Client or their own `DISTRIBUTOR` file.

The Provider determined by the `DISTRIBUTOR` file or the IP network that the client using is called the "Candidate Provider".

6.3.2 Allocation Phases

We distinguish between two "allocation phases". The first is called "pre-login" and the second is the "post-login" phase.

The pre-login phase is before a user has logged in or registered. At this point the user's true Provider is unknown, so the client uses the Candidate Provider (i.e. either the Provider in the `DISTRIBUTOR` file or the Provider associated with the IP network that the client is using) instead.

The post-login phase is after a user has logged in or registered. At this time the user's Provider is fixed. When a user registers, the Candidate Provider becomes permanently associated with the user. So in the post-login phase, the Candidate Provider is irrelevant, and is ignored by the TeamDrive Clients.

However, if the user logs out, he reverts to the "pre-login" phase, and the Candidate Provider is once again associated with the user.

6.4 Provider parameters

As mentioned before, there are a number of Registration Server settings that are associated with a Provider. The settings are described in *Provider Settings* (page 58).

The initial values of these settings are specified in the `RegServerSetup.xml` file (see *Registration Server Settings* (page 51)) when the Provider is first created. After the initial setup, these values can be changed using the Admin Console (see *Administrative Guide*).

6.5 Hosting Service for each Provider

Each Provider can register their own Hosting Service at a Registration Server (only possible with Enterprise Hosting Service). It's also possible to register more than one Hosting Service for the same Provider at a Registration Server, but only one Hosting Service can be used for the default storage accounts of the users for this Provider. You could define your own logic to distribute users to different Hosting Services and use the API to create default storage accounts on the right Hosting Service.

6.6 Client license keys

Each Provider receives their own range of client license keys, which all start with the four letter Provider Code followed by 3 blocks of 4 characters each (ex: `TMDR-1234-1234-1234`). For every user a default license is created (if no global default license is defined, see *DEFAULT_LICENSEKEY* (page 62)). Each license has one or more features which enable actions in the client (for more details, please look at *TeamDrive Client <–> Server interaction* (page 15)).

If a license has an "owner" assigned(who must be an existing user of the licence's provider), then this user will automatically receive the license key when they first install a TeamDrive client. Licenses without an assigned owner (which may be the case for multi-user licenses) can not be automatically assigned (unless it is specified to be the default licence, see *DEFAULT_LICENSEKEY* (page 62)). Instead, a user must manually enter the license code into the TeamDrive Client or have the license assigned to them through the admin console (see "Devices" chapter in administrative guide).

Please note that the owner of a license is not necessarily the same as the user who is using the license. Multi-User licenses will always have users other than the owner. The admin console will show all licenses which are owned and/or used by a user. The admin console also allows you to set the owner of a license or to assign a license from a different owner to existing devices of other users.

License properties:

- `Type`: permanent, monthly payment, yearly payment, not for resale (not possible in the API and Admin Console)
- `Feature`: Banner, allow WebDAV usage, Personal, Professional, Secure-Office (only for mobile Apps)
- `Single`- or ``Multi-User` license. License used will be calculated by user. One user can install and use as many devices with one license

6.7 API access

The Registration Server and Enterprise Hosting Service offer an API interface, so that other systems can execute functions on both systems. The API is using the XML-RPC (http://en.wikipedia.org/wiki/XML-RPC) protocol. For more informations please read the additional API documentation.

Accessing users on the Registration Server using XML-RPC is limited to the users which belong to same Provider. Detecting the Provider depends on the IP address of the request. For each Provider one or more IPs must be enabled. Users which belong to other Provider are not completely invisible, but accessing the email of these users is not possible.

In case that the Registration Server is connected to the TDNS (see *TeamDrive Name Server (TDNS)* (page 29)) a user might already exists on another Registration Server within the TDNS. These users can not be accessed using the API unless the owner of the foreign Registration Server allows API access from you.

CHAPTER

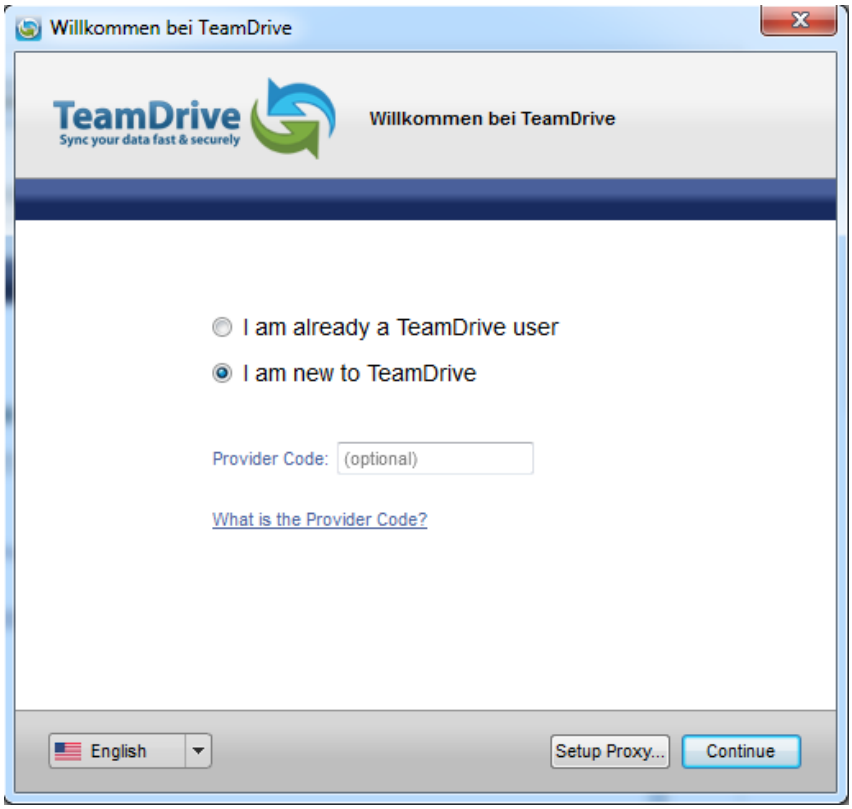
SEVEN

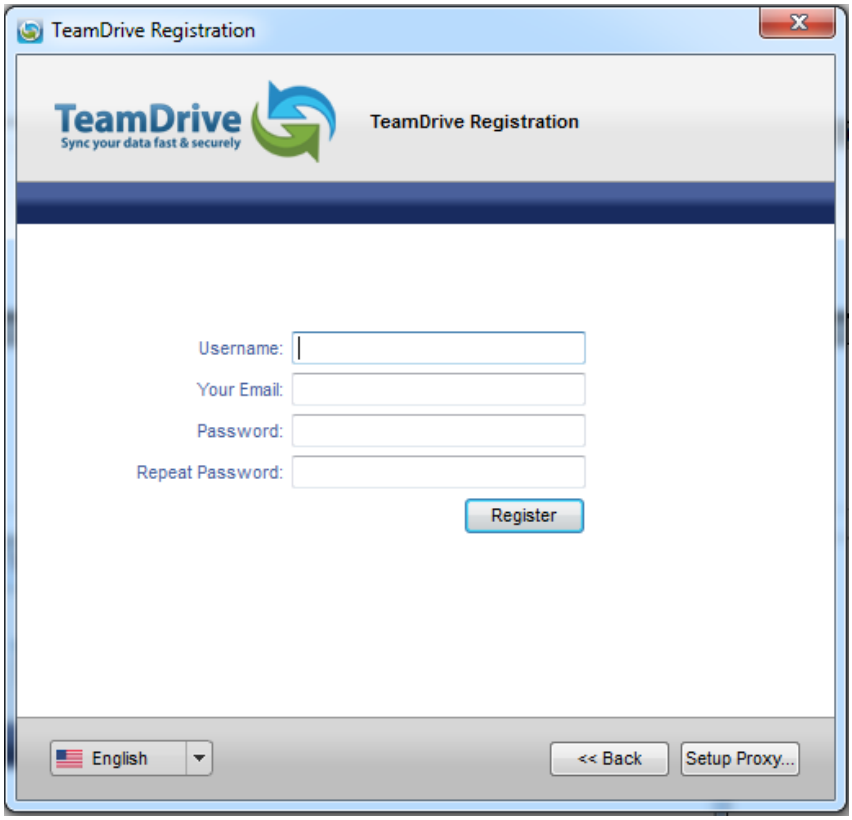
TEAMDRIVE CLIENT <–> SERVER INTERACTION

7.1 User account

7.1.1 Create a new account

A user can use the standard TeamDrive Client to create a new account. So that their account will be registered on the correct Registration Server and for the correct Provider, users will need to enter your provider code. (the first panel of the TeamDrive Client can be suppressed using an OEM TeamDrive Client; please contact TeamDrive Systems for additional information)



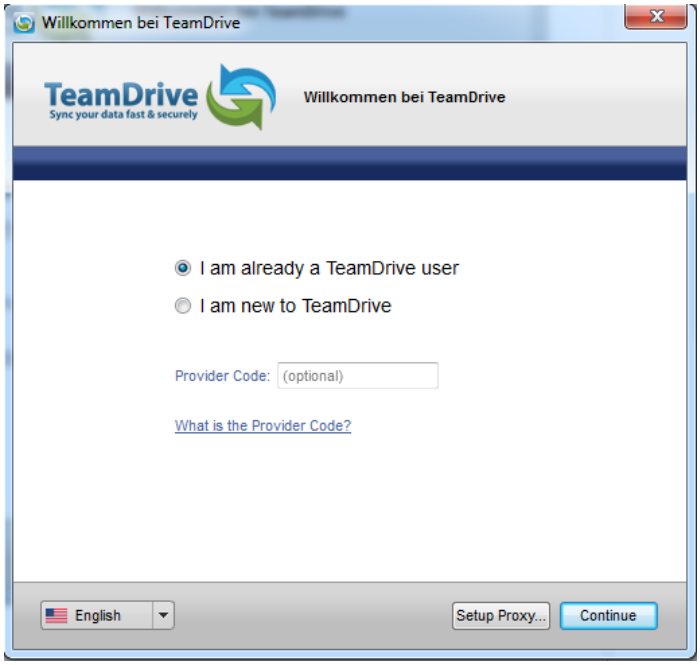


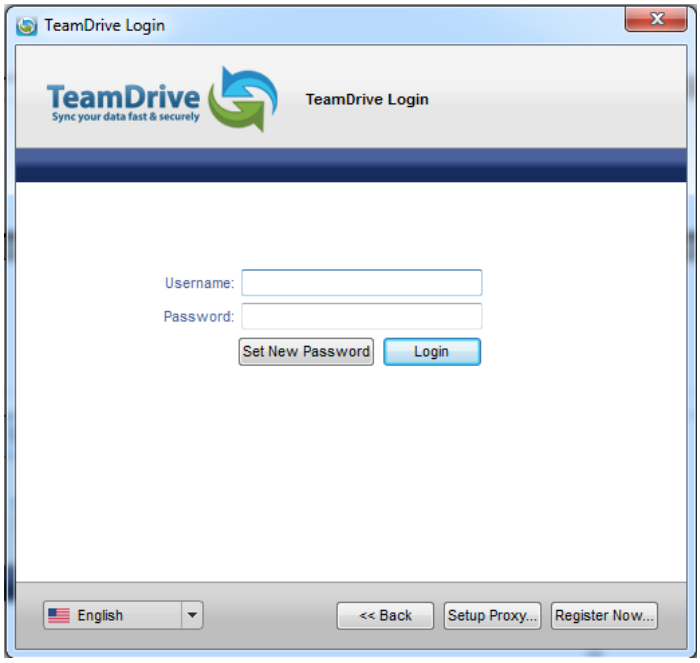
A username, an email address, and a password is required to create an account. As described in *enable-web-login=true/false/default (default: false)* (page 67), you can disable this dialogue and prevent the user from creating a new account using the TeamDrive Client.

Each new account must be activated by an email as described in *Email Templates* (page 26).

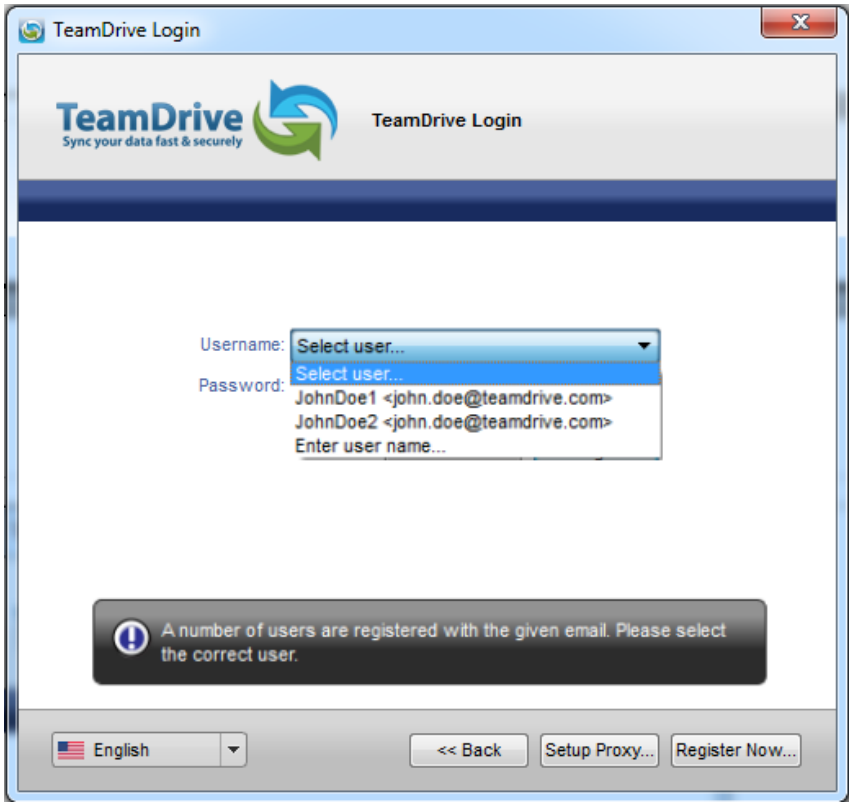
7.1.2 Login to an existing account

If the users already has an account, they can just login and register without typing in the Provider code:





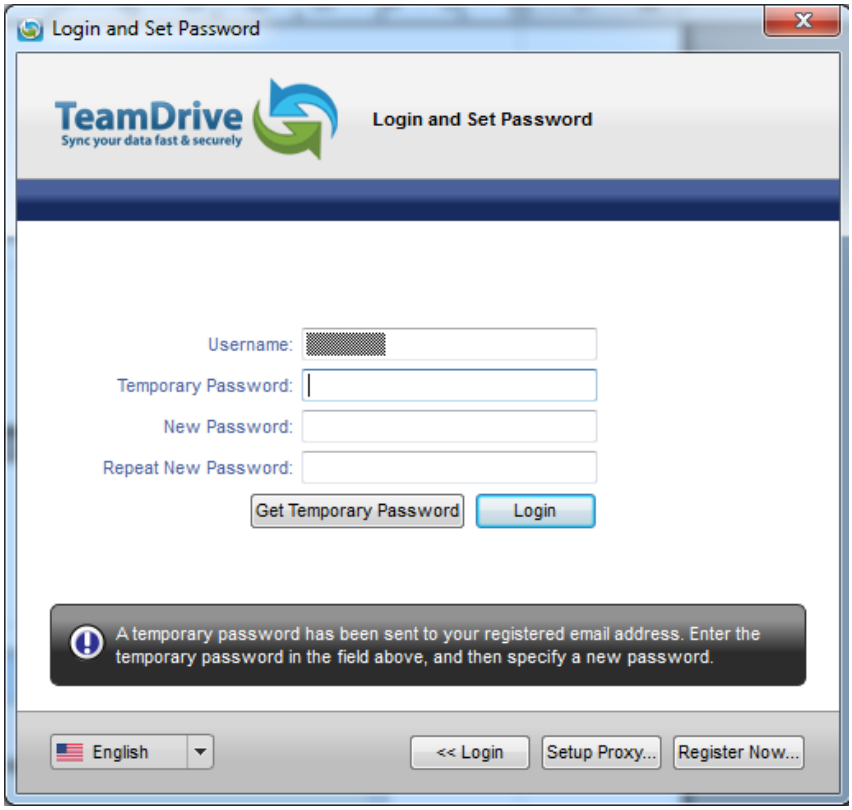
If you enable the setting "`allow-email-login`" as described in *allow-email-login=true/false (default: false)* (page 70) you can also login with an email. If more than one account with that email exists, a list of account usernames and emails will be displayed:



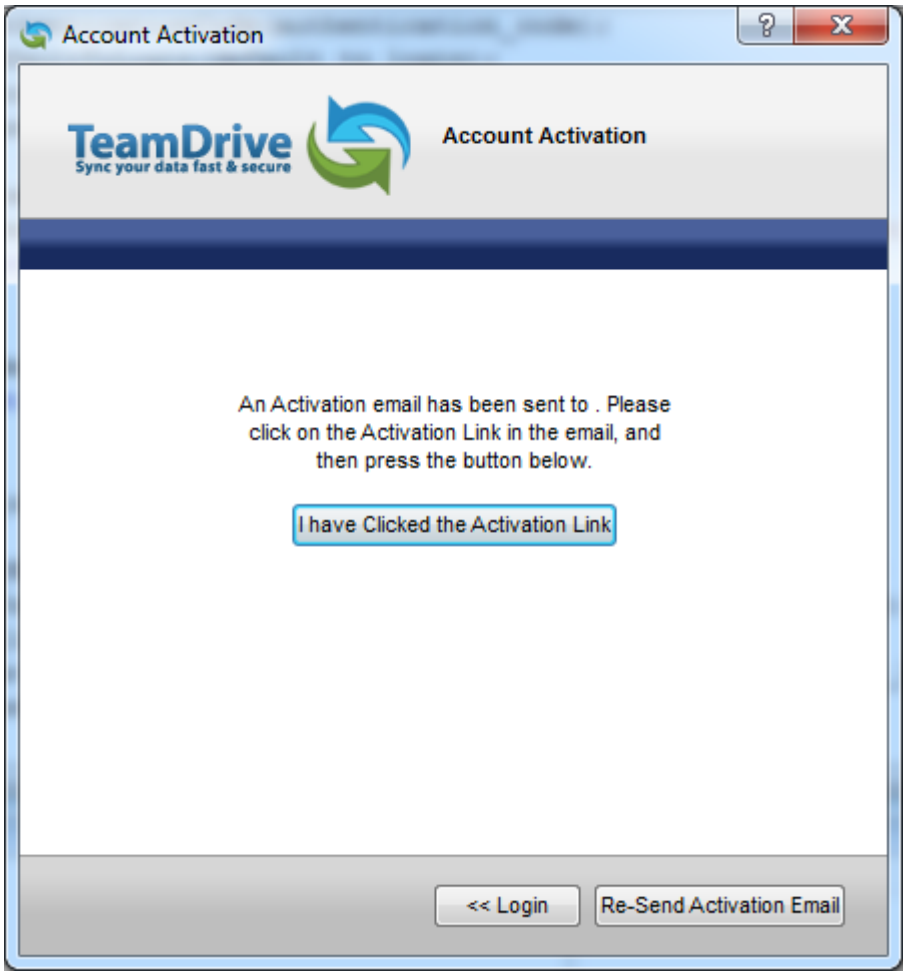
7.1.3 Forgotten password

In case of an unknown* or lost password, the user can set a new password. A temporary password will be send to the users email (as listed on the Registration Server). This temporary password is then entered along with a new password to complete the process.

*This can happen if a user import was performed, as described in the *Administrative Guide*.


Login and Set Password
TeamDrive
Sync your data fast & securely
Login and Set Password
Username:
Temporary Password:
New Password:
Repeat New Password:
Get Temporary Password
Login
A temporary password has been sent to your registered email address. Enter the temporary password in the field above, and then specify a new password.
English
<< Login
Setup Proxy...
Register Now...

7.1.4 Check activation



In order to finish the account creation process, the user needs to click on a link in the activation email (see mail templates in *Templates for Client actions* (page 27)). This behaviour can be modified by the settings described in *Client Settings* (page 56))

7.1.5 Get activation email

The user can click the resend button to resend the activation email.

7.1.6 Undo registration

If the user quits the registration process, the device (see *Device* (page 20)) of the user will be removed.

7.1.7 Retrieve user information

During the registration process the user's data and license will be loaded into the client from the Registration Server in the background. Once users has logged in the email will be retrieved from the Registration Server so it can be displayed in the client. If the user does not have a default license, a new default license will be created for the user depending on the Provider settings (see *DEFAULT_FREE_FEATURE (<FreeClientDefaultLicense> tag)* (page 62)).

7.1.8 Retrieve default storage account on a Hosting Service

This request will ask the Registration Server for a default storage account. Whether a default storage account can be retrieved for this user depends on the Provider settings see (*AutoDistributeDepot* (page 52)). The created account will be stored on the Registration Server, so that the user will get the same storage account again, if they install a second device.

7.2 Device

Each user not only has a user account, but each installation will also create a new owned device on the Registration Server under this user account. The user can install 5 different device types: Mac, Windows, Linux, iOS and Android OS (the amount of devices per user is not limited).

Each device will create its own public-/private key. The public key will be uploaded to the Registration Server for this device. If one users invites another, they will not actually invite the user itself, instead they will invite all the different devices of the target user. This is necessary because each invitation must be encrypted using the public key of the target device.

7.2.1 Invitations

The client will periodically poll the Registration Server for new messages, like invitations. The different types of messages are described in *TeamDrive Client <–> Server interaction* (page 15).

7.2.2 Get public key

If a public key for a device is missing, it will be downloaded from the Registration Server and will be stored in the local key store of the client (filename `PBPG.Keys` in the client user data). In case of another invitation to the same device, the public key from the key store will be used. The keys will be stored under the device id in combination with the Registration Server name, because two different Registration Servers can hold devices with the same id's.

7.2.3 Get device id

Invitations sent will always start with the oldest device of the user. Only active devices from a user can be invited. An active device is defined by the value `<InviteOldDevicesPeriodActive>` as described in *Registration Server Settings* (page 51).

7.3 Messages, Invitations & Invitation Types

All communication between clients is done by sending encrypted messages to the Registration Server which are then retrieved when the server is polled by the receiving client. A message could be an invitation, but other messages types exist and will be described in the following chapters.

7.3.1 Normal invitation

A normal invitation is an invitation to a TeamDrive Space. For further security, invitations can be password protected, requiring the receiving user to enter a password specified by the sender.

7.3.2 Store forward invitation

Existing users can send out Space invitations to users that do not actually have an account on this Registration Server yet by using a "*store forward*"-invitation.

In this case the invitations can not be encrypted using the public key of the target device, because it doesn't exist at this time. Instead, the invitation will be encrypted using the public key of the Registration Server.

If a new user creates an account using the same email address used for the invitation, the background task (see *"Move Store Forward Messages"-Task* (page 49)) will then encrypt the pending invitation using the public key of the newly created device. The new Client then retrieves the invitation within the normal poll request interval.

7.3.3 Invitation for future devices

This functionality was added to resolve the following commonly occurring situation:

> User A installs TeamDrive in his office, creates a few Spaces and fill them with data. At home, he installs TeamDrive on his private PC and expects that he will be able retrieve the data in the Spaces he created in the office.
>
> However, this is not the case because invitations can only be sent to devices with an available public key. Before a device is registered, no public key is available.
>
> User A will need to return to the office, start TeamDrive, and invite himself to all of his Spaces so that his private PC receives and invitation.

To solve this problem, a special invitation is sent for future devices of the user.

To keep this invitation secure, a SHA256 key is generated using the password of the user which is used to encrypt the invitation. Then the encrypted invitation is encrypted again using the Public Key of the Registration Server before being sent.

If the user installs a second device, he will receive this special invitation. The invitation is decrypted using the SHA256 key generated with the users password, allowing the user to access their space with their new device.

Note: This will not work if the user changes his password after the first installation since the client will no longer be able to generate the same SHA256 key that was used to encrypt the invitation.

Future device invitations will be automatically deleted after a specified period of time in seconds. If you set the value to 0 these invitations will never be deleted.

7.3.4 Revoke invitations

An invitation can be revoked by a client. Because all invitations are encrypted and we can not see which invitation might be revoked if the device has more than one invitations stored on the Registration Server, we generate a hash over the Space information. A revoke will remove all invitations which match the hash.

Note: his is only working, if the invitation was not already downloaded by the other client. If that's the case, the user can use the following delete message.

7.3.5 Delete message

Sending a delete message to a user will remove all of their clients for the Space.

7.4 Emails

7.4.1 Invitation email

If an invitation was successfully uploaded to all devices of the invited user, the client will also send an invitation mail. The text for the invitation mail can be modified within the invitation dialogue. It will be send to the Registration Server which will mix the user data with the template of the right Provider and language. The mail templates are described in *Email Templates* (page 26).

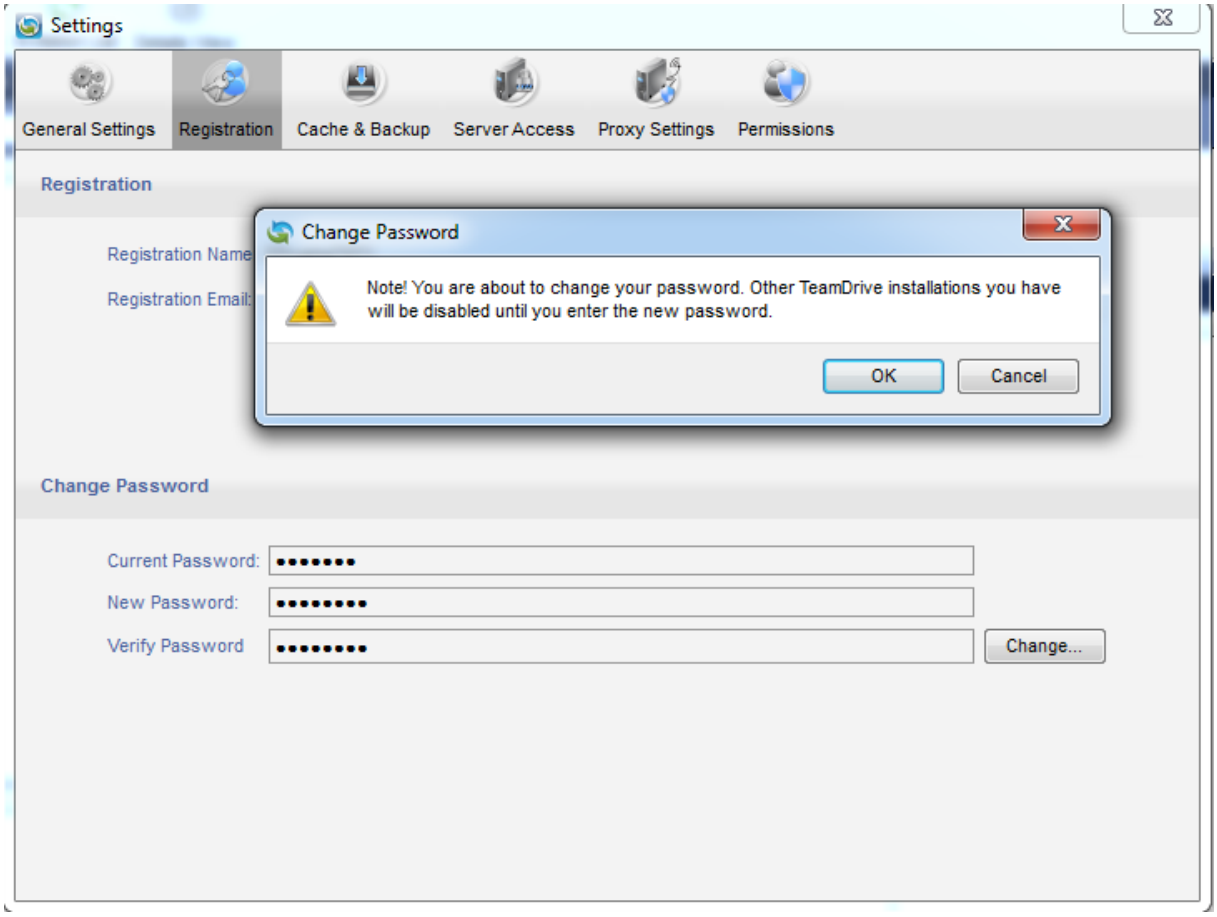
7.4.2 Notification email

The user can send a notification mail to the member(s) of a Space to inform them about changes.

7.5 Change User data

7.5.1 Change password

The user can change their password within the client application (client version `3.0.5.155` required and Registration Server version `3.0.9` required). The current and a new password must be typed in by the user. The new password will be set for all of the user's client installations. Therefore, other installations will prompt the user for their new password with "password has changed" window.

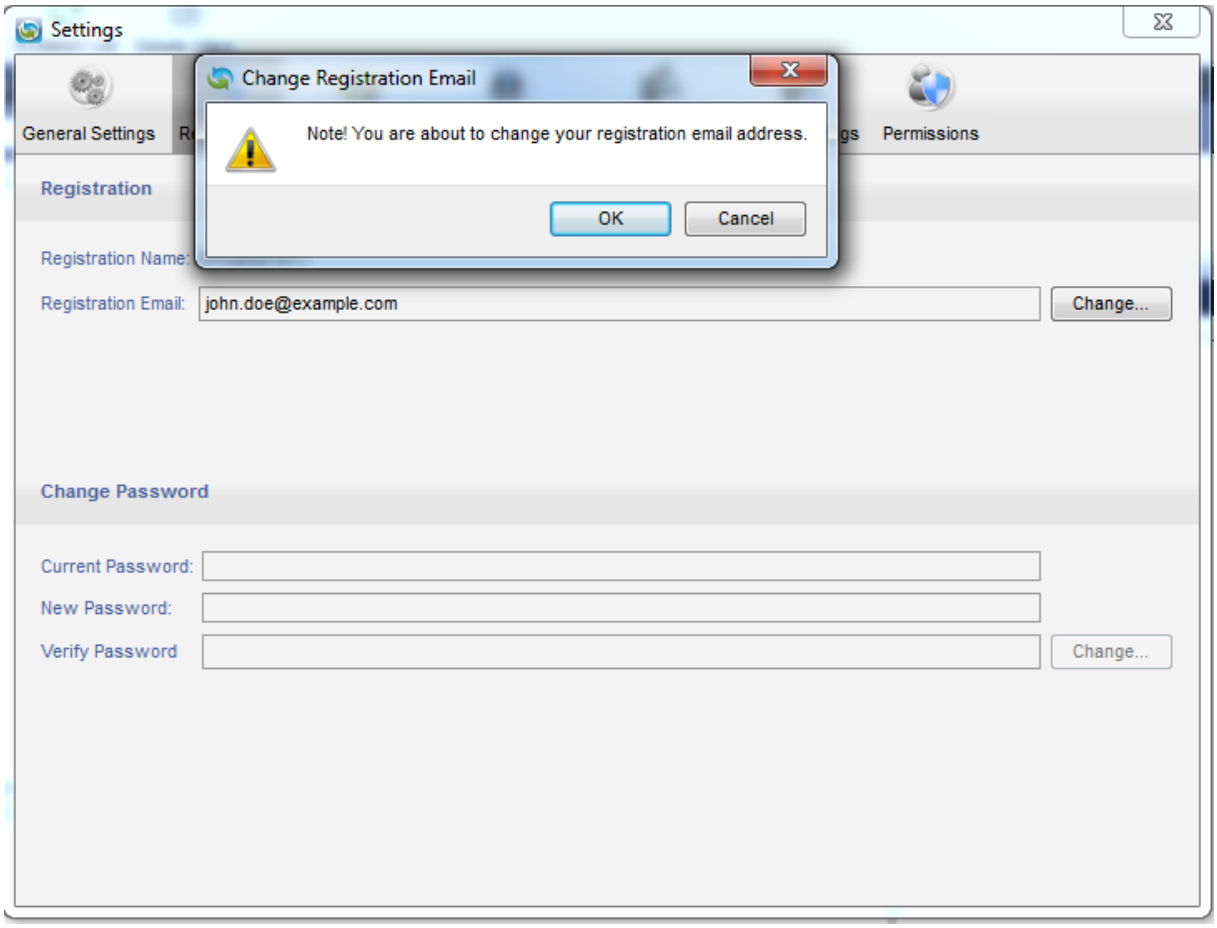


7.5.2 Change email

The user can change their registration email in a two-step process.

1. The user enters their new email address and clicks the *Change* button. This well sent an activation email to the new address.
2. The users clicks the confirmation link with the activation email.

The new email will be changed across all of the user's client installations.



7.5.3 License key

The user can change the license key by manually type in a new key. The key will be changed across all of the user's client installations.

7.6 Banner

The client will compare the locally stored banner with the data on the Registration Server. If the server has new banners available, they will be downloaded and replace the old banner. They will be visible after the client is restarted.

7.7 Updates

The client can be informed about new versions. The user will be informed about an update with a release description. A click on the update button will open a web page where the new version can be downloaded. This is only

available for windows, mac and linux. iOS and Android must be informed about the market place functionality. Updates can be administered from the admin console (see *Administrative Guide*).

7.8 Server URLs

The client will poll in intervals for a new set of URLs. This will not only update the URLs of the own Registration Server, but will also get new informations about all available Registration Server within the TDNS network as described in *"Update RegServer-List"-Task* (page 50)

7.9 Initial Storage Account Request

Unlike the above listed interactions, this request involves both a Registration and a Hosting server.

1. During the registration process the client will ask the Registration Server for a default storage account.
2. The Registration Server will look in his internal database, and check if this user already has a default storage account. If yes, it will be returned. If not, step 3 will be executed.
3. The Registration Server will lookup to which Provider the user belongs and will look for a Hosting Service which belongs to this Provider. A depot creation request for this user will be send to the Hosting Service. The returned value will be stored in the internal database and the result will be also send back to the client.
4. The client is now able to create Spaces on the Hosting Service.

CHAPTER

EIGHT

HTML AND EMAIL TEMPLATES

8.1 HTML Templates

8.1.1 Activation pages

When activating a new TeamDrive installation, an activation link is sent to the user. The activation link will direct them to the Registration Server (or another server that is using the API to activate the device using an API call). Each Provider has their own activation pages, so that they can be modified to the CI of the Provider. The templates are copied to the apache root document during the setup of the Registration Server and when adding an additional Provider. However, they can still be modified afterwards.

A typical path looks like this:

`/var/www/html/activation/tmdr/activation/en/activated_win.html`

The link to the activation pages will be created using the `ActivationURL` (see *ActivationURL* (page 57)), the provider information, the language of the user (if it's part of the allowed languages, see *EMAIL Settings (<Email> XML block)* (page 59) and *ACTIVATION Settings (<Activation> XML block)* (page 60)) and the page itself.

Note: The Provider Code (like "TMDR") in the file system will be converted to lower case in this case.

There are different pages which are nearly the same.

The success page is:

`activated_<platform>.html`

`<platform>` can be *win*, *mac*, *linux*, *ios*, or *android*

Error pages are:

- `already.html`: Link was already clicked and the device is activated
- `error.html`: Unexpected error occurred
- `invalid.html`: Activation code invalid (length != 32)
- `notfound.html`: Activation code not found

8.1.2 Email pages

Changing an email address will send a notification email to the old email address, informing the user the new address is being set for the account, and an activation mail to the new email account. The user must click the activation link in the activation email to confirm the change.

The success page is:

`activated.html`

The error pages:

- `error.html`: Unexpected error occurred
- `invalid.html`: Activation code invalid (length != 32)
- `notfound.html`: Activation code not found

8.2 Email Templates

There are two different template categories. The templates starting with "web-*" in the file name are triggered by actions from the API. All other templates will be triggered by actions from the TeamDrive Client.

There are templates for English and German available. The language in the filename is located at the last part of the filename (example: new-passwd-**de**.email). Additional languages can be added by creating a new file with a new language code.

Each Provider has their own set of templates, so that each Provider can use their own text and graphics in the templates. Each Provider has to define the available and allowed languages in their Provider settings as described in *EMAIL Settings (<Email> XML block)* (page 59).

Templates can be all plain-text or plain-text with an HTML part. By default, the invitation templates have a text and an HTML part. All other templates are completely in plain text. All templates can be modified by you.

The notification mails for spaces or files can not be modified. This mail is directly generated by the teamdrive clients and can not use a template.

8.2.1 Structure of the mail templates

Text mail: The subject of the email will be divided using these two characters "//". Everything before will be used as the subject. Everything behind is the mail body.

HTML mail: The structure is a little more complicated (see http://en.wikipedia.org/wiki/MIME#Multipart_messages), because for mail clients which do not display HTML you have to offer a plain text part. Otherwise the email will be shown as empty within this mail client. The template is divided into several parts. Replace the place holders with your content:

- Definition of a multipart-mail (the boundary string will be used in the following text and HTML part):

  ```
  Content-Type: multipart/alternative; charset=UTF-8;
  boundary='www_teamdrive_net_e_mail_boundary_625141'
  ```

- followed by the subject (divided by "//" again): `//TeamDrive invitation//`
- followed by the text and HTML part:

  ```
  --'www_teamdrive_net_e_mail_boundary_625141'
  Content-Type: text/plain; charset=UTF-8; delsp=yes; format=flowed
  Content-Transfer-Encoding: 8bit

  <Put in your plain text here>

  --'www_teamdrive_net_e_mail_boundary_625141'
  Content-Type: text/html; charset=UTF-8;
  Content-Transfer-Encoding: 8bit

  <put in your HTML code here>

  --'www_teamdrive_net_e_mail_boundary_625141'--
  ```

8.2.2 Templates for Client actions

[[BRAND]] The product brand name, defined in the provider-specific setting `EMAIL/BRAND_NAME`. If not set or empty, the default is "TeamDrive".

`[[FULLGREETING]]`

- **reg-activationlink**: This will send an email with an activation link to the user. They can only proceed with the registration by clicking the link within the email. The link must lead back to your server, so that the activation code can be verified. There are three fields available which will be replaced before the email will be send to the user:

  `[[SERVERURL]]`: This is the URL defined in the xml file as described in *RegServerURL* (page 54). You can also replace it with an other URL which also points to the Registration Server. If you prefer to use an own page, you can use the Registration Server API which can also activate an installation.

  `[[SERVERPATH]]`: The script name ("`pbas/td2as`") of the internal module which handles the activation requests.

  `[[ACTIVATIONCODE]]`: This is the activation code of a non-activated installation. The code is unique for each new installation, and is used for verification by the server.

  `[[DISTRIBUTOR]]`: The Provider Code, which will be used to redirect to the success or error page (which are defined as described in *HTML Templates* (page 25)).

- **reg-notify**: By default, only the first installation must be manually activated (depends on the setting described in *AllowActivationWithoutEmail* (page 56)). The user will just receive a notification mail that an additional device was installed

- **new-passwd**: If the user lost their password, they can reset the password during the login process (see *Forgotten password* (page 17) for details). There must be one field in the email which will be replaced before the email can be send:

  `[[NEWPASSWORD]]`: Only a temporary password will be send, which must be entered in the client together with some new password as specified by the user. Retrieving a new password also depends on the setting as described in *AllowPasswordChangeInClient* (page 57).

  Changing both password and email at the same time is not possible. If the email is different, this has to be changed before the password is changed.

- **reg-emailchangedtonew**: Upon requesting an email change, the user will receive an activation URL to verify that the new email belongs to him. The following fields are available: `[[SERVERURL]]`: The same as described above in `reg-activationlink` `[[SERVERPATH]]`: The same as described above in `reg-activationlink` `[[EMAILVERIFY]]`: An verification code like the activation code in `reg-activationlink` `[[DISTRIBUTOR]]`: The same as described above in `reg-activationlink`

- **reg-emailchangedtoold**: Whenever the user's email is changed, a verification email is sent to the old address (to protect the user against potential hacking attempts). The following fields are available: `[[NEWEMAIL]]`: The new email address of the user

- **td3-privacyinvited-email**: If a new user was invited who currently had no account, they will get an invitation sent to their email by the person who invited the user. A download link for the client application should be in this template so that the user can download and install the client. There are two new fields which have the same content, but have different line breaks: `[[INVITATIONTEXT]]`: The invitation text the user wrote in the client application. Line breaks are *carriage return* `[[INVITATIONTEXTHTML]]`: The same text, but line breaks are HTML conform `<br>` `[[DOWNLOADLINK]]`: Download link taken from the download `Redirect-URL` page as described in *REDIRECT_DOWNLOAD* (page 66).

- **td3-privacyinvitedsecure-email**: Same as above, but with the additional mechanism that the user has to type in a password to accept the invitation. The password will be defined by the user who send the invitation. (This is an additional security option to prevent anyone from accidentally inviting an invalid user)

- **td3-privacyinvited-user**: Nearly the same as an invitation by email, but the user already exists and therefore they get invited via their username.

- **td3-privacyinvitedsecure-user**: Before acceptting the invitation the user must enter a password (as specified by the sender).
- **reg-storageincreasedinvited**: This mail will be used if you use the user referral functionality. Each new user which is invited, as well as the inviter, will get additional storage space. Configuring this functionality is described in *REFERRAL Settings* (page 64).

  This template will be send as a confirmation mail to the user which was invited. You can use the following fields: `[[USERNAME]]`: The username of the invited user `[[REFUSER]]`: The username which invited the new user `[[STORAGEINCREASED]]`: The amount of storage which was added to the account.
- **reg-storageincreasedinviter**: This template will be send as a confirmation mail to the user which invited the new user. You can use the following fields: `[[USERNAME]]`: The username of the user who invited a new user `[[REFUSER]]`: The username of the user which was invited `[[STORAGEINCREASED]]`: The amount of storage which was added to the account.

8.2.3 Templates for API actions

The API also sends emails (this can be disabled, see *API_SEND_EMAIL (<APISendEmail> tag)* (page 59)).

The links within the templates must be point to a page where you call an API function again. For more information see the Registration Server API documentation.

- `web-activationlink`: Nearly the same as described above in `reg-activationlink`.
- `web-delete-user`: Deleting a user will delete all devices. Licenses (if defined) and all Spaces (if defined). So the user has to confirm to delete all his data.
- `web-emailchangedtonew`: Nearly the same as described above in `reg-emailchangedtonew`.
- `web-emailchangedtoold`: Nearly the same as described above in `reg-emailchangedtoold`.
- `web-newpassword`: Nearly the same as described above in `new-passwd`.

CHAPTER

NINE

TEAMDRIVE NAME SERVER (TDNS)

The TeamDrive Name Server (TDNS) allows users from different registration servers to work together by mapping users to their respective registration servers. This allows invitations to be sent to the correct registration server which is necessary because invitations must be send to the Registration Server with which the user registered their devices.

Usernames, unlike email addresses, are unique within the TDNS network. If you enable TDNS, any username registered on an existing Registration Server can not be registered/used on your Registration Server.

TDNS access will modify the registration, login, search and invitation calls in the Registration Server (and also the API calls) and check the TDNS, determining which username exists on which Registration Server in the TDNS network.

Every Provider requires a record on the TDNS. A record will have a *ServerID* and a *checksum*. All requests will contain the *ServerID* and *checksum* to verify that the request is coming from a valid Registration Server.

You have to enable outgoing access on the HTTP-Port 80 to `tdns.teamdrive.net` to enable the communication from your Registration Server to the global TDNS.

9.1 Data security on the TDNS

On the TDNS we don't store usernames or emails in plain text. All data will be hashed and salted in your Registration Server, so that we have only strings like:

UserName `000095C3FE7F65D8F800BAEE55A5BD01`

Email `7F236FD1B733B8E1A2355977AA98D9C5`

This method ensures that no plain usernames and emails will leave your Registration Server. Access to the TDNS is only possible for a Registration Server. The client does not directly access the TDNS.

9.2 Communication workflow from Client to Registration Server to TDNS and the way back

Inviting users which are registered on different Registration Server will result in a couple of requests. Please keep the following facts in mind:

- A client can only poll his own Registration Server for new invitations. Clients registered on other Registration Server must send the invitation to the Registration Server holding the invited client record
- Only the client's own Registration Server can check whether the access credentials of the client are still valid

User is searching for john.doe@example.com:

A) Client -> Search request -> Registration Server 1 -> Hash lookup for the email -> TDNS (List of 3 Registration Servers) -> Registration Server 1 -> Answer to client with the Registration Server list -> Client

B) Request 1 -> Get username for email -> Registration Server 2 -> returning username to client

C) Request 2 -> Get username for email -> Registration Server 3 -> returning username to client

D) Request 3 -> Get username for email -> Registration Server 4 -> returning username to client

Client will show 3 different usernames with the same email in the invitation dialogue. The user will choose the user from Registration Server 2 on the list

E) Client -> Invitation request -> Registration Server 2

Description of the request steps:

A) The user entered an email address in his client and clicks on *add*. A search request will be send to Registration Server 1. Registration Server 1 is converting the characters below `ASCII 127` in the email to lowercase and generates the hash. A lookup will be send to the TNDS. The TDNS will answer with:

- No Registration Server: Email is unknown -> Store forward invitation using the email
- one Registration Server: Email is only registered on one Registration Server -> If the name of the Registration Server is identical, the Registration Server will directly return the username.
- more than one Registration Server: Email is registered on more than one Registration Server; this case will be described in the next request descriptions

B) – D) The client get a list of Registration Server names. The client must now send a search request to each of the Registration Server. The client will send an additional flag, so that no new TDNS lookup will be done. otherwise another list of Registration Servers would be returned. The answers from the different Registration Server will be put together and displayed in one result in the invitation dialogue.

E) After the user has picked the correct user from the list, the invitation will be send to Registration Server 2, where the target user is registered.

It is only possible to connect to other Registration Servers using a special remote authentication. Normally only the own Registration Server can check the authentication. When connecting to a foreign Registration Server, the own Registration Server will create a remote authentication sequence which can be checked by another Registration Server which doesn't know the user.

For you, as a provider of a Registration Server, it's important that you can control which other Registration Server in the TDNS network you trust and which other server you allow your clients to contact. This is done using a black and white-list in the admin console (see *Administrative Guide*).

CHAPTER

TEN

EXTERNAL AUTHENTICATION

TeamDrive supports external authentication. If used, the authentication data is not located on the Registration Server. The TeamDrive Client, version 3.1.1 or later, provides an alternative login window in the form of an embedded browser. This embedded browser-based login window resides in a different panel than the standard login dialogue. By default, this panel is disabled, and must be enabled explicitly by the Client Settings sent from the Registration Server. This procedure is described in detail below.

External authentication is performed by a Web-site, possibly just a single page. This Web-site is called an "Authentication Service". Upon a successful login, the Authentication Service returns a page containing an "Authentication Token". This token is received by the TeamDrive Client and sent to the Registration Server. The Registration Server then uses a pre-defined URL to verify the Authentication Token. Upon successful verification the login will be completed.

10.1 External User Data

In order to complete the login of an externally authenticated user, the Registration Server requires a user ID and the email address of the user.

10.1.1 User ID

A vital prerequisite for the for external authentication is a unique fixed user ID. The Authentication Service **must** provide a unique ID for every user that can be authenticated by the service. Furthermore, the user ID must be fixed (always remain associated with that user) the moment it is first used to identify a user.

The user ID may be any character sequence up to 100 unicode characters (or 300 ascii characters) in length. The character sequence used as the user ID is an internal reference which will not be exposed to the user. This means the character sequence can be cryptic (i.e. it does not need to make sense to the user). The most important characteristics of the user ID is that it's unique and fixed.

Most systems to do not have a problem providing a unique identifier for a user. For example, the email address of the user is globally unique, and can used as the user ID. Many authentication systems, such as LDAP, store a "username", which uniquely identifies the user.

However, some systems have a problem with the "fixed" property of the user ID. For example, the email address of a user can be used as the user ID, but there are situations in which a company may want to change a user's email address.

In general, if the user ID of a user changes, the Registration Server will not recognize a user as the same user when the user logs in for a second time. For example, if a user owns two devices, and the user's ID changes after login on the first device, the Registration Server will consider the login on the second device to be from a different user, even though the user used the same credentials to login on both devices.

10.1.2 Email Address

Users that have been externally authenticated will be identified in the TeamDrive Client by their email address. Invitations sent to externally authenticated users must also use the users email address.

The user's email address may change if it is not used as the user ID. In this case, TeamDrive will only discover the change when the user logs in again. It is possible to force the user to re-login, however this cannot be done automatically since the Registration Server has no way of knowing that a user's email was changed within the Authentication Service.

Re-login is described in *Compelling Re-login* (page 32).

The user's profile name can be used as an alternative to displaying the user's email address in the TeamDrive Client (see *display-full-name=true/false (default: false)* (page 70)).

If the username is hidden, we recommend setting the user's profile information during the external login process, if possible. This is described in *Authentication Examples* (page 33).

10.2 Compelling Re-login

You may need to compel the user to re-login for a number of reasons:

- Updating user information (email address and other profile information) stored by the TeamDrive Client or the Registration Server. Currently, re-logging is the only way to update this information.
- The user's password has changed.
- Confirming the user's identity for security reasons (usually done periodically)

Forcing the user to re-login is currently a manual process. It is done by generating a random MD5 value and updating the column `MD5Password` in the table `TD2Device` (see *TD2Device table* (page 40)) and `TD2User` (see *TD2User table* (page 46)), in the `td2reg` database. All devices and the user row must be set to the same MD5 value.

A random MD5 value is generated by applying the MD5 hash to a randomly generated character sequence. A different random MD5 value should be used for each user.

The result of this update is that the TeamDrive Client on all devices of the user will automatically request login.

10.3 Login Configuration

Login is configured by the client-side settings. The settings that can be used are described in *Login and Registration Client Settings* (page 67). Since the user is in the pre-login phase, the settings used are determined by the Candidate Provider (see *Network Allocation* (page 12)).

If external authentication is required then the embedded browser-based login panel must be enabled. This is done by setting the `enable-web-login` setting (see *enable-web-login=true/false/default (default: false)* (page 67)) to "`true`" or "`default`". If the standard login panel is not disabled (see *enable-login=true/false/default (default: true)* (page 67)) then `enable-web-login` should be set to "`default`". This will ensure that the user is presented with the web login panel when the TeamDrive Client is started.

Next the `AUTH_LOGIN_URL` setting for the Provider must be set to the URL of the page that will handle the authentication. This URL will be called as soon as the web login panel is displayed to the user.

10.4 Lost Password and Registration

Embedded browser-based panels are also available in the login dialogue to preform the "Lost Password" and "Registration" functions.

Their Configuration is similar to the configuration of the Web-based login function. The client-side settings user are `enable-lost-password`, `enable-web-lost-password`, `enable-registration` and `enable-web-registration`, as described in *Login and Registration Client Settings* (page 67).

If you implement these functions then you must set `AUTH_LOST_PWD_URL` and `AUTH_REGISTER_URL` to the corresponding URLs (see *EMAIL Settings (<Email> XML block)* (page 59)).

All your embedded web-pages can be linked together in way expected by the user. For example, the Login page should provide a link to the Lost Password page, if available. Hidden fields (described below) in these pages inform the TeamDrive Client that a page change has occurred, so that the page will be displayed in the correct panel.

Back buttons do not need to be provided by the web-pages. This operation can be performed by using the standard buttons available in the login dialogue.

10.5 Authentication Examples

If you plan to implement your own Authentication Service, please request the example authentication web-sites from TeamDrive Systems GmbH.

We provide two example authentication web-sites.

10.5.1 Demo Authentication

This is a set of PHP pages which provide a simple example of all authentication functions: login, lost password and registration. This code is provided for demonstration purpose only, and should **not** be used in a production environment.

10.5.2 LDAP Authentication

This is an implementation of LDAP Authentication in PHP, meant for reference. This implementation uses the PEAR "Auth" object (http://pear.php.net/package/Auth/docs). You may use these web-pages without modification if you wish to provide LDAP based authentication for your TeamDrive users.

See the chapter "*Configuring External Authentication using Microsoft Active Directory / LDAP*" in the *TeamDrive Registration Server Administration Guide* for details.

The `Auth_Container_LDAP` Auth "container" is used to access the LDAP server and verify the user's credentials.

PEAR Auth provides other containers for many authentication methods, including:

- `Auth_Container_IMAP` – Authenticate against an IMAP server
- `Auth_Container_KADM5` – Authenticate against a Kerberos 5 server
- `Auth_Container_POP3` – Authenticate against a POP3 server
- `Auth_Container_SAP` – Authenticate against a SAP server

The LDAP example can be easily adapted to use one of these alternative authentication methods.

10.6 Authentication Tokens and Verification Pages

As mentioned above, after the Authentication Service has confirmed a user's credentials, it returns an Authentication Token to the TeamDrive Client. The client then sends the token to the Registration Server in order to complete the login.

Before it can successfully complete the login process, the Registration Server must verify the Authentication Token. This is done using the URL stored in the `VERIFY_AUTH_TOKEN_URL` setting (see *VERIFY_AUTH_TOKEN_URL (<VerifyAuthTokenURL> tag)* (page 65)). The page referenced by this URL is called the "verification page".

A verification page performs two functions:

- **Validation of the Authentication Token:** The verification page must confirm that the Authentication Token is valid, and was generated by the Authentication Service. Two examples of how this can be done are provided by the Authentication Examples mentioned above.
- **Return User data required to complete Login:** As mentioned above, in order to complete login, the Registration Server requires the user ID and the email address of the user. This information must be returned by the verification page if validation is successful.

The verification page may be either local or remote.

10.6.1 Remote Verification Page

A remote verification page is located on the Authentication Service server. Verification of the Authentication Token requires the Registration Server to open a secure HTTP connection to the Authentication Service.

The Demo Authentication example described above is an example of a remote verification page. The Authentication Token used in the Demo example contains a reference to user data stored by the authentication web-site (i.e. stored by the Authentication Service).

When the verification page is requested by the Registration Server, the page extracts the reference from the Authentication Token and uses it to retrieve the user ID and email address from local storage. The verification page must be located on the Authentication Service server.

10.6.2 Local Verification Page

A local verification page is located in the Registration Server's local network, possibly on the same machine. A local verification page does not require access to the user's data repository (e.g. LDAP server) because all of the information required to verify and complete login are stored within the Authentication Token.

An example of this is provided by the LDAP Authentication example mentioned above. The Authentication Token returned by the LDAP example contains all data needed to verify and complete login in an encrypted form.

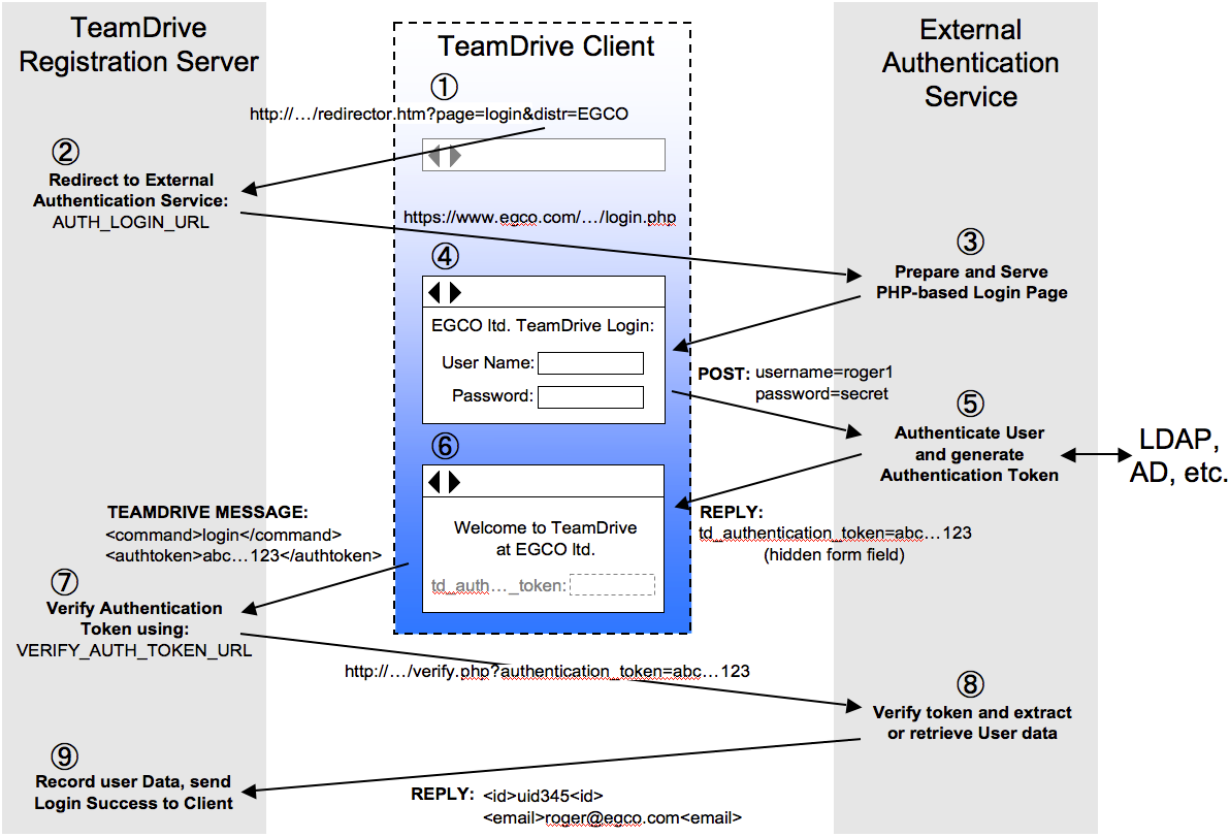
This has the advantage that the Registration Server does not have to connect to the Authentication Service to verify the Authentication Token which may not be possible due to firewalls or for performance reasons.

However, this means that each Authentication Service used must provide a corresponding verification page which can be installed in the Registration Server network. Installation is done by the Registration Server administrator.

Note that a local verification page **must** be implemented in PHP. Other server-side technologies are not supported in order to keep the Registration Server installation as simple as possible.

10.7 Login Procedure

The diagram below illustrates 9 steps that constitute the login procedure. Each step is described in the sections that follow.



10.7.1 TeamDrive Client: Load Registration Server Redirector URL

When the embedded browser-based login panel is displayed, the TeamDrive client loads the redirector URL of the Candidate Provider with the URL parameters: `page` and `distr`. `page` is set to "*login*" and `distr` is set to the Candidate Provider's Provider Code.

10.7.2 Registration Server: Re-direct to AUTH_LOGIN_URL

The Registration Server redirects the client's embedded browser to the `AUTH_LOGIN_URL`. Access to this page must use the secure HTTP protocol (https).

10.7.3 Authentication Service: Generate Login Page

The HTML of the login page is generated and returned to the client by the Authentication Server. The page includes an HTML form with standard fields to gather the user's credentials and perform login.

The HTML form must include the following hidden fields which are evaluated by the TeamDrive Client:

- **td_login_page:**

  For example: <input type="hidden" id="td_login_page" value="login"/>

  This field tells the client which page has been loaded. Possible values are "login", "register" or "lostpassword". The TeamDrive Client will switch the login panel accordingly. In this way, the correct title and buttons will be displayed in the login dialogue.

- **td_registration_server:**

  For example: <input type="hidden" id="td_registration_server" value="TeamDriveMaster"/>

  This field specifies the name of the Registration Server that must be called to complete the login process. This value is actually only needed if the user manually enters an URL in the embedded browser for the login

dialogue. Normally this information is redundant because when the TeamDrive Client loads the page, it has already determined the Candidate Provider and hence the Registration Server.

- **td_distributor_code:**

  For example: <input type="hidden" id="td_distributor_code" value="EGCO"/>

  This field specifies the Provider of the Authentication Service. Just like the `td_registration_server` field above, this field is only required if the user manually enters a URL into the embedded browser in the login dialogue.

  It is recommended that the login page contain elements that make it identifiable as belonging to the Provider. For example by using a logo associated with the Provider.

  This is required because it may not be obvious to the user where he has landed, due to the fact that the identification of the CandidateProvider is transparent to the user. In particular, identification of the Candidate Provider using the current IP address of the client can lead to the user being presented a different login dialogue depending on where the TeamDrive Client is started.

10.7.4 TeamDrive Client: Display Embedded Login Page

The TeamDrive client displays the HTML page received from the Authentication Service.

When the page is loaded, the client also reads the 3 hidden fields described above: `td_login_page`, `td_registration_server` and `td_distributor_code`. Depending on the value of `td_login_page` the client will switch to the appropriate login panel.

If the `td_distributor_code` is set, it may change the Candidate Provider, and is used later, along with the specified Registration Server to complete the login (or registration) process.

10.7.5 Authentication Service: Authenticate User Credentials

When the user clicks the login button, control returns to the Authentication Service's web-site. The target page is determined, as usual, by the page specified in the HTML `<form>` tag.

The Authentication Service then checks the credentials submitted by the user. If an error is encountered, the web-site should return the login page with an appropriate error message.

If the credentials are valid, the returns a page with a message indicating success. As shown in the Authentication Examples (see *Authentication Examples* (page 33)), the result page should also indicate that login is now being processed by the Registration Server.

On success the page must include a form with the following hidden fields:

- **td_authentication_token:**

  For example: <input type="hidden" id="td_authentication_token" value="<?php echo $authToken; ?>"/>

  The authentication token as describe in *Authentication Tokens and Verification Pages* (page 33).

- **td_authentication_cookie:**

  For example: <input type="hidden" id="td_authentication_cookie" value="<?php echo base64_encode($authCookie); ?>"/>

  The authentication cookie is stored by the client. You can store any information you like in the cookie and encrypt the data for security reasons. The cookie is returned by the client when they access the Authentication Service again.

  The cookie should be used to pre-fill the username field when a user is required to re-login. For this purpose it is recommended to store information in the cookie that can be used to identify the user (for example, the user ID).

- **td_user_secret:**

  For example: <input type="hidden" id="td_user_secret" value="<?php echo $userSecret; ?>"/>

  This field is require to support automatic distribution of Space keys to all devices of a particular user. In other words, when a user creates or enters a Space on one device, the "user secret" makes it possible to pass this information securely to all other devices belonging to the user. In particular the access information can be passed securely to devices that are registered later (i.e. devices unknown at the time of Space entry).

  The user secret is optional, but without it, the user must explicitly invite all new devices to his Spaces.

  Note that the user secret is only stored on the TeamDrive Client. In particular, it is not passed to the Registration Server, as this would constituted a security risk (because both encrypted the Space keys and the means to decrypt the keys would be located in the same location).

  For additional security, the client does not use the user secret as is. Instead it uses a salted SHA256 hash value of the user secret.

The result page may also include the following fields which are used to set the user's profile:

- **td_profile_name:** Set the actual name of the user.
- **td_profile_email:** Sets the email address in the user's profile.
- **td_profile_telephone:** Sets the user's telephone number.
- **td_profile_mobile:** Sets the user's mobile phone number.
- **td_profile_notes:** Sets the notes field in the user profile. This field may contain any additional information you wish to distribute regarding the user.

10.7.6 TeamDrive Client: Process Result Page

The TeamDrive Client displays the result page returned by the Authentication Service. If the page contains the `td_authentication_token` then the client assumes that authentication was successful and sends a secure login message to the Registration Server. The login message includes the Authentication Token and the Provider Code of the Candidate Provider.

Other data returned by the Authentication Service is retrieved from the hidden fields in the page and stored locally. This includes the authentication cookie (`td_authentication_cookie`), the user secret (`td_user_secret`), and any profile data sent by the service.

The login dialogue is disabled while the TeamDrive Client waits for a reply from the Registration Server.

10.7.7 Registration Server: Verify Authentication Token

The Registration Server receives the login message from the TeamDrive Client. Using the URL specified by the `VERIFY_AUTH_TOKEN_URL` setting (see *AUTH_VERIFY_PWD_FREQ (<VerifyPasswordFrequency> tag)* (page 65)) it verifies the Authentication Token. The Authentication Token is added as a parameter to the URL with the name "`authentication_token`".

10.7.8 Authentication Service: Execute Verification Page

The page referenced by the `VERIFY_AUTH_TOKEN_URL` setting is called the "verification page". This page verifies the Authentication Token sent by the Registration Server. Further details on how the verification page works are provided in *Authentication Tokens and Verification Pages* (page 33).

The verification page is expected to return the following XML result upon encountering an error:

```
<?xml version='1.0' encoding='UTF-8'?>
<teamdrive>
        <error>
                <message>ERROR_MESSAGE</message>
```

```
        </error>
</teamdrive>
```

The `ERROR_MESSAGE` text will be printed to the Registration Server log, but not returned to the client. Instead the client will display a generic message indicating that authentication failed.

On success, the verification page must send a reply of the following form:

```
<?xml version='1.0' encoding='UTF-8'?>
<teamdrive>
        <user>
                <id>USER_ID</id>
                <email>USER_EMAIL</email>
        </user>
</teamdrive>
```

Here `USER_ID` and `USER_EMAIL` are the values as described in *External User Data* (page 31).

10.7.9 Registration Server: Complete Login

The Registration Server evaluates the XML result sent by the verification page. In general, an error is not expected unless the system has been compromised somehow.

The user ID returned by a successful verification is stored in the `ExtReference` field in the `TD2User` table in the `td2reg` database.

Before inserting a record into the `TD2User` table, the Registration Server checks to see if a user with the given user ID is already present. In this case the user's email address is updated and success is returned to the TeamDrive Client.

If the user ID is not found a new user record is created. Internally, the Registration Server generates a so-called "magic username" for the user. This username is of the form `$DISTCODE-USERCOUNTER`, for example: `$EGCO-1234`.

Magic usernames are never visible to the TeamDrive Client user. Instead, the users e-mail address is used whenever the username would otherwise be displayed or used in the client.

Note: An error will be returned to the client, and login will fail, if the user's email address is already in use by some other user.

CHAPTER

ELEVEN

DATABASES

The Registration Server setup will install 2 databases:

11.1 Database "td2reg"

This is the main database with the following tables:

11.1.1 TD2AddressRange table

Entries to identify users by the ip address (see *Network Allocation* (page 12))

Columns:

- `ID`: Primary key column
- `OwnerID`: Reference to Owner=Table
- `AddressType`: 0 = API Access, 1 = Clients of the Provider
- `MinAddress`: IP address start
- `MaxAddress`: IP address end
- `RangeSize`: IP range

11.1.2 TD2Autotask table

Entries for period tasks (see *Auto Tasks* (page 49) for details)

Columns:

- `ID`: Primary key column
- `Name`: Name of the task
- `Status`: En= or disabled
- `Description`: Task description
- `RunnerConnID`: ID of the PBAC which is processing the task currently to avoid duplicate execution at the same time by two different PBAC
- `RunnerLoginTime`: Start time
- `LastStartTime`: Start time
- `LastEndTime`: End time
- `LastResult`: Error description
- `ProcedureText`: Function call

- `Frequency`: Frequency in seconds, minutes or hours for tasks which should not run on every PBAC start

11.1.3 TD2BlobData table

Stores blob data which is necessary for update notifications for the clients.

Columns:

- `ID`: Primary key column
- `OwnerID`: Reference to Owner=Table
- `CreationTime`: Creation timestamp
- `ModifyTime`: Modification timestamp
- `Type`: type of the blob. Could be "update", "banner", "email", "csv", "logs". Please notice, that not all types are in use at the moment. A few types are planned for future release.
- `IsActive`: Is entry activated or not
- `Language`: Language for the entry to support banner or update informations in different languages
- `Name`: An internal name for the entry
- `Extension`: Additional field to store non blob information for the entry
- `Data`: The file itself

11.1.4 TD2Depots table

Will store the reference to the depots for a user (see Hosting Server Documentation for more info).

Columns:

- `ID`: Primary key column
- `UserID`: Reference to User=Table
- `ReposDoc`: Repository document
- `HostUrl`: URL of the Host=Server
- `RepositoryID`: Repository ID
- `IsDefault`: Default repository or not

11.1.5 TD2Device table

Each client installation will be stored in this table

Columns:

- `ID`: Primary key column
- `MD5Password`: Identical to user md5 password
- `Status`: Status of the device
- `User`: Reference to User=Table
- `Name`: Name of the device
- `Ticket`: Reference to `Ticket=Table`
- `TicketStatus`: Status of a ticket
- `ActivationCode`: Activation code for device activation

- `PublicKey`: Public key of the device
- `CreationTime`: Creation timestamp
- `ActiveTime`: Last active timestamp
- `IPAddress`: IP address of the device when it was created
- `ClientVersion`: Version of the client installation
- `Platform`: Windows, Mac, Linux, iOS, Android
- `ScreenResolution`: For later use
- `PublicKey2`: The new public key using a RSA 2048 Bit key length

11.1.6 TD2Email table

All emails generated by the clients are stored in this table. One autotask process will send the emails from this table

Columns:

- `ID`: Primary key column
- `OwnerID`: Reference to Owner=Table
- `Status`: Status of the mail
- `DestUserID`: Reference to User=Table
- `ErrorMessage`: Error message
- `Destination`: Reference to Device or User=Table (depends on the Template=column)
- `Email`: Destination Email
- `CreationTime`: Creation timestamp
- `SendTime`: Send timestamp
- `Template`: Template name
- `Content`: Mail body in case it's mixed with user data

11.1.7 TD2FreeUserStorage table

If the referral functionality is enabled, information about additional storage space awarded will be stored here (see *REFERRAL Settings* (page 64) for details).

Columns:

- `UserID`: Reference to *User=Table*‘
- `InviteFriend`: Additional free storage due to invited users
- `FreeOffset`: An additional free offset (used for old TeamDrive 2 clients)
- `Fixed`: A fixed value (no changes in `FreeOffset`)

11.1.8 TD2LicenceType table

License types for monthly, yearly, permanent, or not=for=resale licenses

Columns:

- `ID`: Primary key column
- `LicenceName`: License name: Permanent, Monthly Payment, Not for Resale, Yearly Payment

11.1.9 TD2Message table

This table stores invitations and other messages exchanged between Clients until a Client polls for its messages (see *Messages, Invitations & Invitation Types* (page 20))

Columns:

- `ID`: Primary key column
- `Destination`: Reference to Device table. The device which should receive the message
- `Origin`: Reference to Device table. The device which send the message
- `CreationTime`: Creation timestamp
- `Hash`: Hash to identify a message for the revoke function
- `Content`: Encrypted message body

11.1.10 TD2MessageFD table

This table stores invitations for devices of a user which will be installed in the future (see *Invitation for future devices* (page 21))

Columns:

- `ID`: Primary key column
- `User`: Reference to the User=Table
- `Origin`: Reference to the Device table
- `CreationTime`: Creation timestamp
- `Hash`: Hash to identify a message for the revoke function
- `Content`: Encrypted message body
- `ClientVersion`: Version of the client

11.1.11 TD2MessageSF table

This table contains store-forward invitations for new users which don't have an account yet, but have been invited by an existing user to join one of his Spaces (see *Store forward invitation* (page 21))

Columns:

- `ID`: Primary key column
- `Email`: Email of the invited user
- `Origin`: Reference to the Device table
- `CreationTime`: Creation timestamp
- `Hash`: Hash to identify a message for the revoke function
- `Content`: Encrypted message body
- `ClientVersion`: Version of the client

11.1.12 TD2Owner table

The table for the Provider entries (see *Provider Concept* (page 11))

Columns:

- `ID`: Primary key column

- `LoginName`: Loginname
- `MD5Password`: MD5 Password
- `Language`: Language of the Provider
- `FirstName`: Firstname of the Provider
- `LastName`: Lastname of the Provider
- `Email`: Email of the Provider
- `WorkPhone`: Phone number of the Provider
- `CreationDate`: Creation timestamp
- `Gender`: Gender m(ale)/f(emale)
- `Address`: Address of the Provider
- `City`: City of the Provider
- `PostalCode`: PostalCode of the Provider
- `Country`: Country of the Provider
- `Company`: Company of the Provider
- `TicketPrefix`: 4 letter Provider Code
- `LicenseEmail`: CC=Email for license creation

11.1.13 TD2OwnerLicenceType table

Which Provider can use which license types for host licences

Columns:

- `IDOwner`: Reference to Owner table
- `IDLicenceType`: Reference to LicenseType table

11.1.14 TD2OwnerMeta table

A key=value table to store different Provider settings as described in (see *Provider Settings* (page 58))

Columns:

- `ID`: Primary key column
- `OwnerID`: Reference to Owner table
- `Name`: Name of the key
- `Value`: Value of the key

11.1.15 TD2OwnerMetaSetting table

A template table to store all available settings which could be chosen as a new parameter for an entry in `TD2OwnerMeta`.

Columns:

- `ID`: Primary key column
- `Grouping`: A group entry
- `Name`: Name of the key

- `DefaultValue`: A default value of the key. Will be used if a new entry will be created in `TD2OwnerMeta` or if the key in `TD2OwnerMeta` is missing.
- `Description`: A description for this value
- `ReadOnly`: Is the entry read only when editing a `TD2OwnerMeta` value
- `Visible`: Is the entry visible
- `IsUnique`: Is only one entry per provider in `TD2OwnerMeta` allowed
- `Deletable`: Is the entry in `TD2OwnerMeta` deletable
- `Type`: Data type
- `Format`: A regular expression to validate the user input
- `CreationTime`: Creation timestamp
- `ModifyTime`: Modification timestamp

11.1.16 TD2OwnerProduct table

Which Provider can create which licenses for the client and/or TDPS

Columns:

- `IDOwner`: Reference to Owner table
- `IDProduct`: Reference to Product table

11.1.17 TD2Parcel table

The key repository table to store space access data for the users on the server.

Columns:

- `ID`: Primary key column
- `UserID`: Reference to the `TD2User` table
- `Type`: user=private=key, user=public=key or space=key
- `Status`: 1 = active, 2 = archived, 3 = backed up
- `CreationTime`: Creation timestamp
- `ModifyTime`: Modification timestamp
- `PublicKeyID`: Reference to the "user=public=key"=type for a user in this table
- `GlobalID`: An internal global ID
- `Content`: The key data

11.1.18 TD2Product table

The list of products which are available to create licenses

Columns:

- `ID`: Primary key column
- `ProductName`: Name of the product (TeamDrive Client, TeamDrive Personal Server)

11.1.19 TD2RegServerList table

Table with available Registration Server within the TDNS. List will be updated by an autotask process from a server of TeamDrive Systems GmbH. Only active if TDNS usages is activated see *TeamDrive Name Server (TDNS)* (page 29))

Columns:

- `ID`: Primary key column
- `RegServerName`: Name of the Registration Server
- `CreationTime`: Creation timestamp
- `ModifyTime`: Modification timestamp
- `Enabled`: White= or blacklisted
- `Data`: Data block
- `AuthSequence`: Authorization sequence

11.1.20 TD2Setting table

A key=value table to store different global settings for the server as described in *Registration Server Settings* (page 51).

Columns:

- `ID`: Primary key column
- `Grouping`: Field to group values in the GUI
- `Name`: Name of the key
- `Value`: Value of the key
- `Description`: Description of the field
- `ReadOnly`: Readonly attribute
- `Visible`: Visible attribute
- `Type`: Data type (boolean, varchar, integer)
- `Format`: A regular expression to validate the user input
- `CreationTime`: Creation timestamp
- `ModifyTime`: Modification timestamp

11.1.21 TD2Ticket table

The license keys for the clients

Columns:

- `ID`: Primary key column
- `Number`: License number
- `MD5Password`: MD5Password to allow administration of the license by the user
- `Type`: Reference to LicenseType table
- `Status`: Status of the license
- `Owner`: Reference to Owner table
- `ContractNumber`: Free text field

- `HolderEmail`: License email
- `Language`: Language
- `TimeLimit`: Time limitation for the license
- `ValidUntil`: A fixed valid timestamp
- `LicenseLimit`: Single or Multi=User license value
- `LicenseUsed`: License in use
- `FeatureFlag`: Combination of Banner, WebDAV, Personal and/or Professional
- `CreationTime`: Creation timestamp
- `UserID`: Reference to User table if license is bound to a user
- `Product`: TeamDrive Client or TDPS
- `IsDefault`: Default license
- `ExtReference`: Free text field to store an external reference

11.1.22 TD2TicketChanges table

A history table with all changes made to license keys (amount of seats changed or changed features)

Columns:

- `ID`: Primary key column
- `Ticket`: Reference to Ticket table
- `WhatChanged`: Information what changed
- `ChangeDate`: Change date
- `ChangeID`: Text field with change information
- `Status`: The original `status`
- `ContractNumber`: The original `ContractNumber`
- `HolderEmail`: The original `HolderEmail`
- `ValidUntil`: The original `ValidUntil`
- `LicenseLimit`: The original `LicenseLimit`
- `FeatureFlag`: The original `FeatureFlag`
- `LicenseKey`: The original `LicenseKey`

11.1.23 TD2User table

A table with all registered users

Columns:

- `ID`: Primary key column
- `CreationTime`: Creation timestamp
- `UserName`: Username
- `TdnsUserName`: The hashed Username for TDNS
- `Email`: Email
- `ExtReference`: Free text field to store an external reference

- `Department`: Free text field
- `ExtAuthID`: External ID. Will be set when using external authentication (see *External Authentication* (page 31))
- `AllowEmailSearch`: Not longer used
- `MD5Password`: Salted MD5Password of the user
- `Language`: Language of the user
- `EmailDay`: Date when the last mail was sent
- `EmailCount`: How many mails were send by this user a the `EmailDay`
- `TmpPwd`: Temporary password for lost password function
- `TmpEmail`: New Email until email is confirmed by the link
- `TmpSession`: Used during registration process
- `Status`: Status of the user
- `InvitedBy`: Reference to the user which invited this user
- `OwnerID`: Reference to Owner table
- `Capability`: User settings to enable or disable certain functionality

11.1.24 TD2UserPrivileges table

Assigned privileges for users for the access to the Admin Console

Columns:

- `userId`
- `flagList`

11.1.25 TD2UserPrivilegesSetting table

Available privileges for users for the access to the Admin Console

Columns:

- `ID`
- `Grouping`
- `Name`
- `Description`
- `CreationTime`
- `ModifyTime`

11.1.26 Keys table

This table stores the all keys for the Registration Server. The private and public key of this Registration Server and all public keys of the clients will be stored in this database. Prior to version 3.0.017, this table was stored in a separate database named "pbpg".

Columns:

- `Code`: Primary key column
- `UniqueDevice`: Unique devicename within the TDNS network

- `Version`: Public key version
- `CreationTime`: Creation timestamp
- `Data`: Public key of the device

11.2 Database "td2apilog"

A database with one "`TD2APIRequests`"=table. API calls can be logged in this table to identify problems when using the Registration Server API interface (look in *Registration Server Settings* (page 51) for the part about request logging).

Columns:

- `ID`: Primary key column
- `Created`: Creation timestamp
- `IPAddress`: IP of the request
- `Command`: API function call name
- `User`: Username
- `Request`: XML from the request
- `Answer`: XML which was send back

CHAPTER

TWELVE

AUTO TASKS

There is a number of background jobs that are being performed by the PBAC-based `teamdrive` service.

You can review and manage them via the Registration Server Administration Console by clicking **Manage Auto Tasks**. See chapter *Managing Auto Tasks* in the *TeamDrive Registration Server Administration Guide* for details.

The frequency of how often the PBAC will wake up can be changed by editing the file `/usr/local/primebase/pbstab`. The `-i`-Option defines the interval in seconds. The default value is 10 seconds.

Note that the frequency of the individual tasks can be defined differently, by changing the task's **Frequency** setting.

12.1 "Send Emails"-Task

This process sends out emails generated by the Team Drive Clients (e.g. device activation or Space invitation messages), which are queued in the Registration Server's internal email queue.

12.2 "Delete Old Messages"-Task

Messages not retrieved by Clients will be deleted from the Registration Server's internal message queues after the period defined in the Registration Server settings `<InvitationStoragePeriod>` (e.g. invitations and other client messages, store-forward invitations) and `<InvitationStoragePeriodFD>` (invitations for future devices)

See *Registration Server Settings* (page 51) and *Messages, Invitations & Invitation Types* (page 20) for details on these settings and the various message types.

12.3 "Move Store Forward Messages"-Task

Existing users can send out Space invitations to users that do not actually have an account on this Registration Server yet.

This background task encrypts pending invitations for newly registered users using the public key of the newly created device and enqueues them in the regular message queue. The new Client then retrieves the invitation within the normal poll request interval.

See *Store forward invitation* (page 21) for details.

12.4 "Delete Client IPs"-Task

For privacy/data protection reasons, this task removes the Client IP addresses from the Devices table according to the value of `<StoreRegistrationDeviceIPinSeconds>` as described in *Registration Server Settings* (page 51).

12.5 "Update RegServer-List"-Task

If TDNS access is active, this task will poll the TeamDrive Master Registration Server to retrieve a list of new Registration Servers within the TDNS network.

Users registered on your Registration Server can only invite users from white listed Registration Servers to their Spaces.

By default, this will be performed every 12 hours.

The automatic white listing of servers depends on the setting `<TDNSAutoWhiteList>`.

- If set to `true`, new Registration Servers will be automatically white listed.
- If set to `false` you have to approve each Registration Server manually, via the **Manage Servers** page of the Administration Console.

12.6 "CleanUp"-Task

If API logging is enabled (the global setting `APIRequestLogging` is set to `True`), each API request is logged in a database table. On a busy server, these log entries can significantly increase the size of the Registration Server's database over time.

Enabling this task will remove entries from the API log table, if they are older than 30 days. This task is disabled by default.

12.7 "Expire Licenses"-Task

If you issue licenses with an expiration date (by entering a date in the **Valid until** field), this task takes care of sending out reminder emails to the license's user(s), informing them about the upcoming expiration. Once the expiration date has been reached, the license will be invalidated and the user's TeamDrive clients will fall back to their default license.

CHAPTER

THIRTEEN

SETTINGS

13.1 Registration Server Settings

The Registration Server Settings are configured using the XML file (`RegServerSetup.xml`) during the initial setup of the server. Afterwards, these settings can be changed in the Admin Console, via the **Edit Settings** page.

The names of the settings below are identical to the names of the tags in the XML configuration file.

13.1.1 General Settings

AllowManageLicence

Set to **True** to enable license administration via the Administration Console.

APIAllowSettingDistributor

If you are using the Admin Console and have multiple providers using the registration server, then this value must be set to **True**.

APIChecksumSalt

To detect *man in the middle* attacks when sending API requests to your server, you can define a salt. The sender must do a MD5 hash of the content which will be send and add the salt to his request. The checksum will be send in the URL, so that the Registration Server can check if the content was modified during the transport. For details, look the API documentation. If the left empty during the initial installation, the setup script will create a checksum salt automatically.

APILogFile

Log file that tracks API requests issued by the Administration Console. This file needs to be owned and writeable by the apache user. (default: `/var/log/td-adminconsole-api.log`)

APIRequestLogging

The Registration Server has an API interface, which uses a the XML-RPC interface (see http://en.wikipedia.org/wiki/XML-RPC for details). The API functions are described in the API documentation. Using this switch you can enable api call logging for all calls to the database (to help identify problems). The requests will be logged in the database `td2apilog` in the table `TD2APIRequests`.

AutoDistributeDepot

If you connect a Hosting Service to your Registration Server, the clients will request a default storage account on a Hosting Service. Each user has one default storage account. It's possible to add more storage accounts to users but only the default storage account can be retrieved by the clients. Enabling the `AutoDistributeDepot` will also distribute the other accounts which belongs to the user to a new client installation.

BalanceURL

On startup and later on in intervals between 4 and 12 hours, the client will ask for new `RegServerURL`s using the balance call (if the value is empty, the `RegServerURL` is used instead). If more than one URL is returned, the client uses a round robin algorithm to send the request to all different servers. The first URL in the list is the URL which must be always available. This will be used as a fallback URL, if the others in the list fail.

CacheIntervall

Cache Interval in seconds. Changes in `TD2Setting` and `TD2OwnerMeta` will be reloaded after `CacheIntervall` expired.

ClientPollInterval

The default poll interval for clients (in seconds) to look for new invitations on the Registration Server.

ClientSettings

These settings are sent to all Clients after login. Settings specified for a Provider can override the values defined here.

DefaultDistributor

One Provider/Distributor which is registered at the server must be the default Provider (this is normally the first provider). For more informations about the Provider concept, please have a look at *Provider Concept* (page 11).

HOSTProxyPort

TCP port of the HTTP proxy server to be used for Host Server API requests.

HOSTProxyURL

IP address or host name of the HTTP proxy server to be used for Host Server API requests.

HOSTUseProxy

Set to **True** if outgoing Host Server API requests must use a HTTP proxy. This requires setting `HostProxyURL` and `HostProxyPort` as well.

HTDocsDirectory

The path to the apaches htdocs directory. During the setup the script will move HTML pages which are necessary for activation and other functions to the right location. The path depends on your 4 letter Provider Code (see *Provider Concept* (page 11) and *HTML and EMail Templates* (page 25) for details)

InvitationStoragePeriod

Invitations will be stored on the server for a specified period of time. The default is 30 days (2592000 seconds). After that duration the server will automatically delete older invitations. If the value is to 0, invitations will never be deleted. Deletions are carried out by the background task described here: *"Delete Old Messages"-Task* (page 49).

InvitationStoragePeriodFD

Within 14 days after the first registration, the client will send an invitation for each created Space to the registration server for devices the user may install in future. See *Invitation for future devices* (page 21) for a detailed description.

InviteOldDevicesPeriodActive

Each new installation by a user will create a new device in the database. If the user were to get a new PC, it would be installed as a new device, but the first device will remain in the Registration Server database even if the user no longer uses it. Invitations will only be send to devices which were active within the defined period. Please notice, that the device active timestamp will only be updated once a day. So, the value should not be less than one day (86400 seconds). The default value is 96 days (8294400 seconds).

LogUploadURL

In the case of a problem in the client, the client can upload its log files to a server. The log will be send to a PHP script. We recommend using our URL since in general it will only be possible for TeamDrive Systems GmbH to understand the log output. If you want to use your own log upload server, please contact TeamDrive Systems to get the upload script and receive further information to access log files.

Note: The Registration Server is running in apache worker mode, which will start several threads within one apache process to answer the requests. Because PHP is not thread-safe, it might be not possible to use the same apache for log uploads. It depends on whether your linux distribution offers ZEND Thread Safety or uses the Fast CGI PHP configuration.

MasterServerName

The name of the Master Registration Server in your TeamDrive Network.

MediaURL

For TeamDrive clients using the free version, the media URL is used to download new banners from the server, which are to be displayed.

NotificationURL

It's possible to send notifications to other Space members with the client. The notification URL will be used to send these notifications. Please use the same domain as in `RegServerURL`.

PingURL

For an inital connection or later on the online test, the client will ping the `PingURL`. This will return a defined answer:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <intresult>0</intresult>
</teamdrive>
```

back to the client, so that the client can check if he can reach the server, or if there is a proxy or an other gateway which require additional steps to get internet access. The `PingURL` can be located on another server and just requires a file `ping.xml` with the above content. Default should be the same domain as in `RegServerURL`,

RedirectorURL

This URL will be used by the client to open web pages in response to clicks within the client. The client will add different parameters to the URL, so that the same URL can be used to call different pages. The `RedirectorURL` will not open the expected target page directly, it will redirect the call to the right target page. Therefore it's possible to change the URLs later in the admin console without changing the client. There are different target pages defined:

- `ProviderInfoURL`: Info page about available provider codes
- `ForumURL`: A link to a forum
- `TutorialURL`: Link to tutorials
- `FAQURL`: Link to FAQs
- `TDPSOrderURL`: Link to buy a license for a TDPS
- `LicensePurchaseURL`: Buy client licenses
- `DownloadURL`: A link to your client download page

These links could be changed using the admin console in the global settings for all provider or for each provider individual in the Provider settings (see *Provider Settings* (page 58)).

RegServerAPIURL

Optional Reg Server API URL, used by the Administration Console (e.g. `http://regserver.yourdomain.com/pbas/td2as/api/api.htm`). Must be set, if HTTPS should be used for API communication or if a dedicated API server is used. If empty, it will be derived from `RegServerURL`.

RegServerName

This is the name of the Registration Server. The name must be unique within the TDNS network. The name must be defined together with TeamDrive Systems GmbH. The name can not be changed later on without reinstalling *all* clients.

RegServerURL

This is the main URL which will be used by the Clients to register and interact with the Registration Server. This URL must always be reachable by the Clients to offer the services. If the URL is not longer valid the Clients have no possibility to reach the server again.

ServerTimeZone

Timezone used for date functions in the Adminstration Console. Please ensure that the timezone is valid (see `/usr/share/zoneinfo/` for available time zone information)! (default: `Europe/Berlin`)

SimulateRegServer20

Enables backward compatibility with TeamDrive 2 clients.

StoreRegistrationDeviceIPinSeconds

Each client registration will store the IP address which was used to register the client. In case of a hacked account, it may be possible to identify the source of the request. The default is 2592000 seconds (30 days) after which the IP will be removed. Other possible values are -1 (never store the value) or 0 (never delete it). All values greater than zero will be taken as seconds. The **Delete Client IPs** auto task as described in *"Delete Client IPs"-Task* (page 50) must be enabled.

TDNSAutoWhiteList

Each new Registration Server in the TDNS network will be added to the "`TD2RegServerList`" table and must be white listed manually. You can define an automatic whitelisting of new servers using this field.

TDNSEnabled

This value will be used to activate the TDNS integration of the RegServer, so that the users of your Registration Server can invite users of other Registration Servers which are registered in the TDNS network. Each Registration Server in the TDNS needs a `ServerID` and a Checksum value which will be defined by TeamDrive Systems. Without these values your server can not communicate with the TDNS. The two values must later be set in the `<TDNS>` section for each Provider (see *TDNS Settings (<TDNS> XML block)* (page 60)).

TDNSProxyPort

TCP Port of the HTTP proxy server to be used for submitting TDNS requests.

TDNSProxyURL

IP address or host name of the HTTP proxy to be used for submitting TDNS requests.

TDNSUseProxy

Set to **True** if outgoing TDNS requests must be sent via a proxy. This requires setting `TDNSProxyURL` and `TDNSProxyPort` as well.

UpdateAvailableURL

The Clients will call this URL to retrieve update notifications about newer versions. The server will return a HTML file with update information which is then displayed in the client. Upon clicking update, the client will open a browser and direct the user to your download page. The update notification can be configured using the admin console (see *Administrative Guide*).

UserNameCaseInsensitive

Set to `$false` if usernames should be case sensitive. By default usernames are case insensitive. Since case-sensitive usernames can be a security risk, this is the recommended setting.

13.1.2 Mail Configuration

MailHeaderHeloHost

As described in the SMTP protocol http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol#SMTP_transport_example there will be communication between the SMTP client on the registration server and the SMTP server which will accept the email for delivery. To avoid spam classification the `HELO` command must match the servers `FQDN`. If this value is empty, the default hostname / IP address detection will be used which might get `127.0.0.1` instead of the hostname.

MailHeaderSenderEmail

The sender header can be defined to avoid spam classification (see sender field description in: http://en.wikipedia.org/wiki/Email#Header_fields). This is necessary in case that the invitations between the users doesn't match to the domain which will be used by the registration server. If this value is empty, only the from header will be used.

MaxEmailPerDay

This is a security setting, since invitation mails can, potentially, also be used for spam mails from an user sent by your mail server. You can define how many mails the user can send per day.

PathToEMailtemplates

This is the start path to the templates. Each Provider will have their own templates. During the setup, the files will be copied into a directory using your 4 letter Provider Code.

SMTPServer

The IP or DNS name of the SMTP server. It must be a SMTP server which can receive `sendmail` requests without require an authentication.

SMTPServerTimeOut

Timeout parameter in seconds for `sendmail` requests

UseSenderEmail

This setting will control, if the above `InvitationEmailReplyTo` will be used or the email of the user which sends the email.

13.1.3 Client Settings

AllowActivationWithoutEmail

If `false`, additional devices (after the first device was successfully activated) must be activated by an email. If `true`, additional devices are automatically activated. In this case the user will receive a notification mail instead of the activation mail.

AllowEmailChangeInClient

If another system will handle the user data, email changes should be disabled for the client, to avoid differences between the two systems. You have to use the API functions to synchronize email changes in your system back to the Registration Server and the TeamDrive Client. To disable the dialogue in the client a modification in the client is also necessary.

AllowNewRegistration

This setting controls whether users can create new user accounts on the Registration Server. Disable this setting if your users were imported into the Registration Server or if direct access to an alternative authentication system is used. Client settings must be used to disable the registration panels on the Client.

AllowPasswordChangeInClient

Very similar to `AllowEmailChangeInClient`. If another system controls the user records, this should also be disabled. To disable the dialogue in the client there a modification in the client is also necessary.

ClientPasswordLength

You can define a minimum password length to be used by a user. The default value is 8 characters. This parameter will only be checked by the API, since the Clients only send an MD5 hash of the password, which can not be checked on server side. A password complexity check is not implemented at the moment.

ClientUsernameLength

You can define a minimum username length to be used by a user. The default value is 5 characters.

UserEmailUnique

Whether the user email is unique so that not two different accounts could use the same email.

Note: This value is only valid per Registration Server. Duplicate emails within the TDNS network are allowed.

13.1.4 Settings required for HTML pages

GlobalActivationSettings

In this block are the settings for the HTML pages for the email activation and email change process defined.

ActivationURL

The activation URL will be used to open a result page when the user clicks on the activation link to activate their account. A detailed description can be found in *HTML and EMail Templates* (page 25).

ActivationHtdocsPath

This will be the start directory below your apache htdocs document root. The complete URL will be generated from the code and will also include the Provider information.

13.1.5 Login Security Settings

The following settings control some security related features used by the Administration Console. They can only be changed via the Administration Console (**Edit Settings -> LoginSecurity**).

FailedLoginLog

Log file that tracks failed login attempts. (default: `/var/log/td-adminconsole-failedlogins.log`)

LoginMaxAttempts

The number of failed login attempts to a particular account within `LoginMaxInterval` before further login attempts are subjected to a delay. (default: `5`)

LoginMaxInterval

Time interval used by `LoginMaxAttempts`, in minutes. (default: `60`)

LoginSessionTimeout

Period of idle time before you need to log in to the Admin Console again, in minutes. (default: `30`)

13.2 Provider Settings

When a Provider (formerly known as a "Distributor") is created, the Provider settings are specified in an XML file in the `<Distributor>` XML block. The Provider is then created using this XML file. After this point, the Provider settings can be changed using the Admin Console.

In the sections below each setting is identified by the name used in the Admin Console. The associated XML tag used during setup follows the name in parenthesis.

To secure the login to the Admin Console you could setup a single IP in the Admin Console using the `LOGIN_IP` setting. Only logins from this IP address are allowed (that also provide correct user login credentials).

13.2.1 API Settings (<APIAccess> XML block)

You can define whether a Provider is allowed to access the Registration Server using the API interface as described in the API documentation.

If the `<APIAccessEnabled>` tag is set to `$false` all settings in this section will be ignored. NOTE: This setting must be enabled to use the Admin Console.

API_IP_ACCESS (<APIAccessIP> tag)

API access is only allowed from a single IP address. Two different providers can not use the same IP address, because the IP address will be used to identify the Provider. This is done for security reasons, as you may only access your own customers, licenses, and other data belonging to your Provider.

Exception: The Admin Console (which uses API functions) allows each Provider to login if the above value `<APIAllowSettingDistributor>` is enabled. It's also possible to add more than one IP address if you want to access the API from different machines.

API_REDIRECT (<APIRedirect> tag)

This value will be returned for various API calls if the calling user belongs to another Provider. For more details, please look in the API documentation.

API_CREATE_DEFAULT_LICENSE (<CreateDefaultLicense> tag)

By default each user will get a default license. You can disable this if you want the possibility of creating or assigning a license created by an API call.

REG_NAME_COMPLEXITY (<RegNameComplexity> tag)

This value must be identical to the value set in the `DISTRIBUTOR file`. For further details, see *reg-name-complexity (default: basic-ascii)* (page 69).

API_SEND_EMAIL (<APISendEmail> tag)

If set to‘‘$true‘‘, the user will be sent an activation email if registered via the API.

API_ALLOW_CHECKSUMERR

When developing or testing the API functions it may be easier to disable checksum checks. This value can only be set in the admin console (see *Administrative Guide*).

API_USE_SSL_FOR_HOST

If your host server can communicate using HTTPS, you could enable SSL communication between the admin console and host server API. This value can only be set in the admin console (see *Administrative Guide*).

API_ADMINCONSOLE_LIC_REF

License reference value when creating licenses using the admin console.

13.2.2 EMAIL Settings (<Email> XML block)

EMAIL_SENDER_EMAIL (<RegistrationSenderEmail> tag)

The activation mail will list this email address as the sender.

EMAIL_REPLYTO (<InvitationEmailReplyTo> tag)

This address will be used for invitation mails. Its usage depends on the value in `<UseSenderEmail>`.

EMAIL_DEFAULT_LANG (<DefaultEmailLanguage> tag)

If the user is using a language which is not listed in `<AllowedEmailLanguage>`, the `<DefaultEmailLanguage>` will be used instead.

EMAIL_ALLOWED_LANG (<AllowedEmailLanguage> tag)

Each Provider Code defines a comma separated list of languages allowed for the emails. A set of templates is required for each language. The language used depends on the language setting of the user's record.

13.2.3 ACTIVATION Settings (<Activation> XML block)

ACTIVATION_DEFAULT_LANG (<DefaultActivationLanguage> tag)

The activation page's language depends on the chosen language of the user. If the language of the user is not supported, the default language specified here will be used. The default HTML pages must always be available.

ACTIVATION_ALLOWED_LANG (<AllowedActivationLanguage> tag)

This is a comma separated list of languages allowed.

13.2.4 BANNER Settings (<Banner> XML block)

The client can display two different banners. One in the main window and one in the Space creation wizard. The banner feature in the user's license specifies whether a banner is displayed. A default banner will be shipped together with the installation package. Banners can be updated using the Admin Console, see chapter "Manage Banner" ToDo: links setzen. You have to set the `MediaURL` as described in *MediaURL* (page 53) which will be used by the TeamDrive clients to download the banner data.

BANNER_ENABLED (<BannerEnabled> tag)

Specifies whether the Banner update function of the Registration Server is enabled.

BANNER_DEFAULT_LANG (<DefaultBannerLanguage> tag)

Banners depend on the chosen language of the user. If the language of the user is not supported, the default language will be used. This banner must always be available.

BANNER_ALLOWED_LANG (<AllowedBannerLanguage> tag)

This is a comma separated list of allowed banner languages.

13.2.5 TDNS Settings (<TDNS> XML block)

If TDNS access is enabled for the Registration Server, each Provider needs its own `ServerID` and `Checksum`.

TDNS_SERVERID (<TDNSServerID> tag)

The ID of the Provider's entry in the TDNS.

TDNS_CHECKSUMKEY (<TDNSCheckSum> tag)

The checksum which will be added to the checksum over the request which will be send to the TDNS. For more details please look at *TeamDrive Name Server (TDNS)* (page 29).

13.2.6 CLIENT Settings (<Client> XML block)

USE_EMAIL_AS_REFERENCE (<UseEmailAsReference> tag)

Enable if you want to use the email address to reference your users between your own system and the Registration Server. In this case usernames will be automatically generated when using the API. All further API requests which require the username can then use the email instead.

ALLOWED_DIST_CODES (<AllowedDistCodes> tag)

A list of allowed Client Provider Codes, besides the Provider's own code (value specified in the `<TicketPrefix>` tag). This refers to the Provider Code in the TeamDrive Client's `DISTRIBUTOR` file. The default value is '*', which means all codes are allowed. '*.' means all Provider which exists on this Registration Server are allowed.

This setting caters for Provider that have a specific version of the TeamDrive Client and want to ensure that only this type of client is used by the Provider's users. Such versions are identified by the Provider Code specified in the `DISTRIBUTOR` file. Since the `DISTRIBUTOR` file is signed it cannot be manipulated on the client side, and therefore, this value can be trusted.

Note: It is highly recommended that Provider always allows the standard TeamDrive Client (which has the TMDR code) in addition to any others.

CLIENT_DEFAULTLICREF

License reference value for default licenses.

CLIENT_NETWORKS (<ClientNetworks> tag)

This is a list of networks (in CIDR notation) or IP addresses that identify users of the Provider. Using this setting, a Provider can determine that certain networks "belong" to the Provider. For example, any company that has been allocated a Provider Code can take ownership of own networks (as determined by global IP address ranges), and use this fact to control TeamDrive Clients started in those networks.

When a TeamDrive Client connects to the Registration Server, and before the user has logged in, the server determines the client's IP address and checks whether the client is running in a network that has been specifically allocated to a Provider. If so, then the Provider Code is sent to the client and this overrides Provider Code in the `DISTRIBUTOR` file. This way, if the user registers after this point, the user will be automatically allocated to the Provider that owns the network in which the client was started.

PRE_LOGIN_SETTINGS (<PreLoginSettings> tag)

These settings are sent to the TeamDrive Client before login or registration. As a result, they can be used to configure login and registration in the same manner as settings within the `DISTRIBUTOR` file. Settings from the server always override client-side settings, so these settings will also override the values in the `DISTRIBUTOR` file.

The Provider of the user must be ascertained before the pre-login settings can be sent to the client. Before login or registration, the Provider of the user is either determined by the Provider Code in the `DISTRIBUTOR` file or the IP address of the client, if it is found to be in a network belonging to a specific Provider. The IP address has priority over the `DISTRIBUTOR` file.

CLIENT_SETTINGS (<DistClientSettings> tag)

These settings are sent to the client after registration or login. Theses settings can be used to configure the behaviour of the TeamDrive Client as required by the Provider. They will override any settings made on the client-side, and also override the global Registration Server `ClientSettings` setting as describe in *Client Settings* (page 56).

Note that after registration or login, the user's Provider is fixed, and therefore the Provider Code in the `DISTRIBUTOR` file, or the network (see *Client Settings* (page 56)) in which the client is stated doesn't play a role any more.

13.2.7 FREE CLIENT Settings (<FreeClient> XML block)

This block will handle the free client limitations. Because we support both WebDAV servers and the TDPS, it's not enough to use the storage account limitation of the Hosting Service. To limit the amount of data which can be administrated using a non full licensed client, you must define the limits here. The value also depends on the Provider.

DEFAULT_FREE_FEATURE (<FreeClientDefaultLicense> tag)

Numerous features can be bound to a license. The default features are set using this parameter. This value uses a bit-mask for enabling or disabling the individual feature; each feature has an assigned value (which is a power of 2) and the value of this setting is equal to the sum of all enabled feature values:

1 = Banner
2 = WebDAV
4 = Personal
8 = Professional

Example: For TeamDrive Systems, the Banner and WebDAV features are set for a free client. The Banner feature has the value 1 and WebDAV feature has the value 2. So to use both, the parameter value would be 1 + 2 = **3**.

For more details about the features, please have a look at *TeamDrive Client <–> Server interaction* (page 15).

FREE_LIMIT_SIZE (<FreeClientMaxDataSize> tag)

This is the value in bytes to limit the amount of data which can be handled by a free client over all Spaces. The limitation will be shown in the client if he is reaching the 75 % border. A progress bar will be visible right above the status bar in the client. If the user will reach the 100 % he can still synchronize data, but the client is switching to meta data synchronisation. Downloading the contents of the files must be initiated manually by the user for each single file and version.

DEFAULT_LICENSEKEY

This value can only be set in the Admin Console, because the license key must be created after the server was setup. This value overrides the `DEFAULT_FREE_FEATURE` setting (`<FreeClientDefaultLicense>` tag).

13.2.8 UPDATE Settings (<ClientUpdate> XML block)

The TeamDrive Client checks if there are updates available for its version. You can use the following settings to define the supported languages for the update notification. How the update notification will be configured using the Admin Console is described in chapter.

MINIMUM_CLIENT_VERSION (<MinimumClientVersion> tag)

Any clients with a version below this may not register a new device.

UPDATE_DEFAULT_LANG (<AutoUpdateDefaultLanguage> tag)

Which update information HTML page will be displayed for the user, depends on the chosen language of the user. The language of the displayed update information HTML page depends on the user's language. If the language of the user is not supported, the default language specified here will be used. The default HTML pages must always be available.

UPDATE_ALLOWED_LANG (<AutoUpdateAllowedLanguage> tag)

A comma separated list of allowed languages.

UPDATE_TEST_USER

A test user can be defined using the Admin Console. This user will always get the update notification in their client even if they are already using a newer version. This allows you to test the update notification without up- and downgrading a TeamDrive client version.

13.2.9 HOSTSERVER Settings

A TeamDrive Enterprise Host-Server is registered using a provider code and the URL of the Registration Server. You can also use the Admin Console to define a default Host-Server for clients which register using said provider code.

The main provider of a Registration Server is allowed to configure any host-server on the Registration Server to accept users with different provider codes. This way it's possible to use only one host server for multiple providers on a Registration Server.

HOST_SERVER_NAME

Please choose a server from the list to use as the default depot server for new clients.

HOST_SERVER_URL

The URL of the Host-Server will automatically be entered in this field after you have selected a host server from the `HOST_SERVER_NAME` list above.

HOST_DEPOT_SIZE

The size of the depot for the user in bytes. Default is: `2 GB = 2147483648 Bytes`

HOST_TRAFFIC_SIZE

The monthly allowed traffic for the user in bytes. Default is: `20 GB = 21474836480 Bytes`

13.2.10 REFERRAL Settings

You can configure a Referral-Program for users which got invited by an existing user.

Note: A "referral" is only valid if:

1. The invited user did not have an account before being invited
2. The user was invited by email
3. The invited user registered with the email address the invitation was sent to (so that a match can be made)

The background task as described in *"Move Store Forward Messages"-Task* (page 49) will do the matching, increasing the depot values and sending the notification mails (see *Templates for Client actions* (page 27)). The background task must be running and be active. You also need an active host server and default depots for your users (see above *HOSTSERVER Settings* (page 63)).

MAX_PROMOTION_USER

The maximum amount of new users which can be invited by an existing user.

PROMOTION_UPGRADE

The promotions upgrade size in bytes. The depot limit and free client limit are increased for both the new and for the existing user.

13.2.11 CSVIMPORT Settings

The settings for the CSV import can only be set using the Admin Console.

Users can be created using a CSV import. The CSV file is uploaded using the Admin Console and will be stored in the database. A cronjob must be configured (see *Administrative Guide*) so that the uploaded files will processed. The success or error logs can be downloaded using the Admin Console.

For backward compatibility you can still use the upload folder in the file system by defining the following optional CSV folders. The file will be imported into the database and then processed from the database entry.

CSV_IMPORT_ACTIVE

Controls whether CSV import will be used.

CSV_UPLOAD_DIR (optional)

CSV hot folder. If not defined, the CSV processing will just use the database. If defined, the contained files will be imported to the database and processed from the database record. Processed CSV files can be downloaded again from the admin console, if necessary.

CSV_ERROR_DIR (optional)

Error logs for not imported users will be written to this folder. If not defined, you will find the value in the database using the admin console.

CSV_SUCCESS_DIR (optional)

Success logs for imported users will be written to this folder. If not defined, you will find the value in the database using the admin console.

DISABLE_MISSING_CSV_USERS

This value will control, whether users not present in the CSV file are disabled. The additional "department" column must be filled in the CSV file. Username and department must match the existing record in the database.

13.2.12 AUTHSERVICE Settings (<AuthService> XML block)

These settings are used to configure access to an external Authentication Service (see *External Authentication* (page 31)).

When referenced by the TeamDrive Client, all URLs (except `VERIFY_AUTH_TOKEN_URL`) below include the parameters specified for all `REDIRECT URL`s (see below).

USE_AUTH_SERVICE (<AuthServiceEnabled> tag)

Set to `$true` if you want to use an external Authentication Service.

AUTH_LOGIN_URL (<AuthServiceLoginURL> tag)

This URL points to the Login page of the external Authentication Service.

AUTH_REGISTER_URL (<AuthServiceRegisterURL> tag)

This URL points to the Registration page of the external Authentication Service.

AUTH_LOST_PWD_URL (<AuthServiceLostPasswordURL> tag)

This URL points to the Lost Password page of the external Authentication Service.

AUTH_CHANGE_EMAIL_URL (<AuthServiceChangeEmailURL> tag)

This URL points to the Change Email page of the external Authentication Service.

VERIFY_AUTH_TOKEN_URL (<VerifyAuthTokenURL> tag)

This URL is used by the Registration Server to verify an Authentication Token, sent by the client after login using the Authentication Service.

AUTH_VERIFY_PWD_FREQ (<VerifyPasswordFrequency> tag)

Maximum length of time (in minutes) user may remain logged in before they are required to enter their password again. If this value is 0, users are never promoted to re-enter their password.

13.2.13 REDIRECT Settings

The `REDIRECT` settings determine the landing pages reached when links are clicked or activated in the TeamDrive Client. These settings cannot be configured in the XML configuration file when creating a Provider.

The banner URLs are not covered by these settings. The HTML document that defines any given banner specifies where its links lead.

The first stop reached when a link is clicked or activated in the client is the determined by the `RedirectorURL` setting. The Registration Server implementation of this page re-directs the user as specified by the settings below.

A number of URL parameters are passed to the landing pages. These parameters can be used within the target landing pages to generate the content.

- `page` and `distr`: These parameters are used to determine the target page of the re-direct. Normally you will not need to evaluate these parameters because the target page is specified by the settings below.
- `lang`: The international language code of the current language of the client.
- `platf`: Specifies the platform of the client: mac, win, linux, ios, android or unknown.
- `size`: The size of the display area for the requested page: width x height in pixels (ex: `400x500`).
- `user`: Base 64 encoded username. This parameter is only supplied for the `REDIRECT_PURCHASE` URL.
- `product`: Specifies the product ordered. Only provided for the `REDIRECT_ORDER` URL. Currently the only possible value is TDPS.
- `cookie`: This is the cookie stored by the client which was passed to the client after a successful external user authentication (see *Login Procedure* (page 34)).

REDIRECT_DOWNLOAD

This URL redirects to a page where the Provider's version of TeamDrive can be downloaded.

REDIRECT_FORUM

This URL redirects to the Provider's forum page.

REDIRECT_TUTORIALS

This URL redirects to the Provider's tutorials page.

REDIRECT_FAQ

This URL redirects to the Provider's FAQ (frequently asked questions) page.

REDIRECT_PURCHASE

This URL redirects to the Provider's page for ordering licenses (parameter '`user`' is the base 64 encoded user name).

REDIRECT_USERINVITEUSER

This URL redirects to the Provider's `user-invite-user` page.

REDIRECT_ORDER

This URL redirects to the Provider's product order page (parameter '`product`' can currently only be 'tdps').

REDIRECT_UNINSTALL

Will be opened after an uninstall operation to present a survey to the user asking why they uninstalled the software.

13.3 Login and Registration Client Settings

The following settings influence the behaviour of the TeamDrive Client during login and registration. They can be set in the `DISTRIBUTOR` file installed on the Client, or as (pre-)login settings on the Registration Server, by adding them to the following Provider Settings:

`CLIENT_SETTINGS` Client settings which are applied after login (multiple settings must each be placed on a new line).

`PRE_LOGIN_SETTINGS` Client settings which are applied before login (multiple settings must each be placed on a new line).

The following TeamDrive Client settings can be adjusted:

13.3.1 enable-change-email=true/false (default: true)

Whether a user may change his email address in the TeamDrive Client application. If the email address will be determined by another system (e.g. when using external authentication), it may not be appropriate for users to change their email addresses via the TeamDrive Client.

13.3.2 enable-browser-change-email=true/false (default: false)

Whether a user may change his email address using the default web browser on the system. This requires the Provider setting `AUTH_CHANGE_EMAIL_URL` to be defined to point to a web page that supports changing the email address.

13.3.3 enable-set-password=true/false (default: true)

Enables/disables the button to request a new password in the login dialogue. Should be disabled in case that the Registration Server is configured to use external authentication.

13.3.4 enable-login=true/false/default (default: true)

This setting can be used to disable the standard login dialogue. If disabled, you should enable the embedded browser-based login using the `enable-web-login` setting. If both standard and web login are enabled, you can determine standard login to be the default by setting this variable to `default` instead of `true`.

13.3.5 enable-web-login=true/false/default (default: false)

"Web login" refers to the embedded browser-based login used for external authentication. If you wish Clients to use external authentication then you must set this setting to `true`. If both standard login (Registration Server based authentication) and web login are enabled then you can determine web login to be the default by setting this variable to `default`.

13.3.6 enable-registration=true/false/default (default: true)

The registration panel in the login dialogue that allows a user to create a new user account on the Registration Server. If user accounts are created by some other mechanism, then you may want to disable registration from within the TeamDrive Client.

If disabled, User accounts must be created using the Registration Server API or a user import script as described in *importing user accounts via csv files*. Another possibility is the use of an external authentication service that accesses an existing user repository such as an LDAP server or Active Directory (see *External Authentication* (page 31)).

13.3.7 enable-web-registration=true/false/default (default: false)

This variable is used to enable the embedded browser-based registration panel in the TeamDrive Client's login dialogue. This may be desirable if you are using an external authentication system which allows user registration. In this case you must create a web-page which performs the registration as describe in *External Authentication* (page 31).

13.3.8 enable-browser-registration=true/false (default: true)

If both standard web-based registration panels are disabled then the TeamDrive Client will direct the user to a web-page when the registration button is clicked. If you do not have such a web-page, then setting this variable to "`false`" will remove the registration button from the login dialogue.

13.3.9 enable-lost-password=true/false (default: true)

A user can request a new password within the login dialogue. Disable this if the user's passwords are not managed by the Registration Server (for example when using external authentication).

13.3.10 enable-web-lost-password=true/false (default: false)

This enables the embedded browser-based lost password panel in the login dialogue. This allows you to directly connect the lost password functionality with an external authentication service (see *Lost Password and Registration* (page 32)).

13.3.11 enable-browser-lost-password=true/false (default: true)

If the standard and web-based password lost panels are disabled, the TeamDrive Client will direct users to a specified web-page where the user can request a forgotten password. If you do not have such a page, then setting this variable to "`false`" will remove the lost password button from the login dialogue.

13.3.12 enable-provider-panel=true/false (default: false)

Defines if the user should be able to enter a different provider code prior to log in/registration.

13.3.13 require-provider-code=true/false (default: false)

Defines if entering a provider code is required.

13.3.14 fixed-provider-code=true/false (default: false)

Defines if the pre-defined provider code defined by `code` can be overridden.

13.3.15 enable-enterprise-server=true/false (default: true)

You can disable the usage of a Hosting Service.

Note: Only the creation of Spaces using a Hosting Service is disabled. Accepting invitations to a Space which is located on a Hosting Service is always possible.

13.3.16 enable-tdps=true/false (default: true)

You can disable the usage of a TDPS (TeamDrive Personal Server).

Note: Only the creation of Spaces using a TDPS is disabled. Accepting invitations to a Space which is located on a TDPS is always possible.

13.3.17 enable-webdav=true/false (default: true)

You can disable the usage of a WebDAV server.

Note: Only the creation of Spaces using a WebDAV server is disabled. Accepting invitations to a Space which is located on a WebDAV server is always possible.

13.3.18 enable-import-server=true/false (default: true)

Defines whether WebDAV, TDPS or Host-Server Depot files can be imported into the client.

13.3.19 reg-name-complexity (default: basic-ascii)

The Registration Server supports all UTF-8 characters for an username. If the Registration Server is connected to an alternative authentication system it might be necessary to limit the allowed characters.

You can assign these values:

- `basic-ascii` (default): A-Z, a-z, 0-9, _, -, .
- `non-space-ascii`: All ASCII characters between code 32 and 127 are allowed
- `printable-unicode`: All printable characters as described here: http://qt-project.org/doc/qt-4.8/qchar.html#isPrint
- `all-unicode`: All characters

If you use one of these values in the `DISTRIBUTOR` file and are using the Registration Server API, then you need to assign the same value for the API access (see *REG_NAME_COMPLEXITY (<RegNameComplexity> tag)* (page 59)).

13.3.20 require-profile=true/false (default: false)

Once created, the TeamDrive username can not be changed. Users can delete their old accounts and create new ones, however, they will not be able not rejoin their previous Spaces without receiving a new invitation for each one of them.

If you want to use external authentication or import users from an external source (as described in chapter *importing user accounts via csv files*), you might want to use the username as a reference field for your system. This can be a database ID from your system. To avoid that this ID will be displayed in the TeamDrive Client as the username, you can set "`require-profile`" or "`profile-uses-reg-email`" to `true`. Users will fill in a personal user profile during the registration process. These profiles only exists within the client.

Space-specific profiles also exist for each Space in the normal TeamDrive Client. They can be used to define different profile data for each Space, however, they are optional.

The "`require-profile`" setting will *require* users to use the profile functionality from the beginning of the installation.

Important: If the username field is being used to store a database ID, users must be allowed to log in with their

email address as opposed to their username (which is unknown to them). Enable "`allow-email-login`" for this purpose.

13.3.21 allow-email-login=true/false (default: false)

In case of using an external reference in the username field as described in *require-profile=true/false (default: false)* (page 69), you should allow email login. Note that the email field is not unique in TeamDrive. If the same email address is used by different accounts, the client will show a drop-down list of all possible accounts after the email address was entered.

13.3.22 display-full-name=true/false (default: false)

Should only be enabled in combination with "`require-profile`" as described in *require-profile=true/false (default: false)* (page 69). If the value is set to true, the client will show the profile name instead of the username.

13.3.23 spaces-path

Default path for newly created Spaces by the user.

13.3.24 check-for-updates=true/false (default: true)

The TeamDrive Client will check for software updates on the Registration Server. Set this value to false, if a software distribution tool will be used to deploy Client installations to your users.

13.3.25 auto-accept-invitation=true/false (default: false)

When set to true, the TeamDrive Client will accept all Space invitations automatically and join these Spaces.

13.3.26 auto-accept-invitation-mode (default: archived)

The mode of operation when joining Spaces automatically. Possible values are: current-version-only, all-versions, directory-information-only, no-sync-to-filesystem, offline-available, archived.

13.3.27 auto-invite-users=list

A list of user names to be automatically invited into newly created Spaces with specified DefaultInvitationRights. The list has to be separated by semicolons and enclosed with double quotes in the settings file. Example: auto-invite-users="abc;def"

13.3.28 master-user=username

A single unique user name that will automatically be invited into every newly created Space with the MasterUserRights privilege. The user must already exist with at least one activated device.

13.3.29 scan-enabled=true/false (default: true)

The internal database will be compared with the file system using a file system scan to detect Space changes while TeamDrive was not running.

13.3.30 hash-compare-files=true/false (default: false)

If set to false, TeamDrive will only use file size and the timestamp to detect new versions. Advantage: Scanning will be faster for spaces with big files. Disadvantage: New versions might be created in case that an application changes the timestamp without modifying the content of the file.

13.3.31 allow-store-forward-invitations=true/false (default: true)

Invitations to users that do not exist, will be invited using a store forward invitation. The user must register with the same email address the invitation was sent to. Should be disabled if the Registration Server does not allow external users to register or in case that the email will be used as username, which might be a problem if the users cannot distinguish between known and unknown users. In the case of an unknown user, the client will automatically send a store forward invitation if the username looks like an email address.

13.3.32 enable-key-repository=true/false (default: true)

Enable/disable the Key Repository.

CHAPTER

FOURTEEN

REGISTRATION SERVER API

14.1 API Basics

The TeamDrive Enterprise Server architecture provides an extensive application programming interface (API) that can be used to:

- Script/automate processes that would otherwise require use of the web-based administration console
- Obtain information about various entities and parameters (e.g. user names, licenses, storage).

The API is based on XML Remote Procedure Calls (see http://en.wikipedia.org/wiki/XML-RPC for a detailed description). Only HTTP POST-Requests will be accepted. Each request must include a checksum in the URL appended as a parameter. This checksum is created by calculating a MD5 checksum over the request body appended with a server-specific salt value. The checksum value must be provided in lower case characters, e.g. by passing it through the `tolower()` function of the respective programming language.

On the TeamDrive Registration Server Administration Console, this salt value can be obtained from the `APIChecksumSalt` system setting (*"Edit Settings -> RegServer"*). On a TeamDrive Host Server, this value is stored in the configuration setting `API_SALT` and must match the value of the Registration Server this Host Server has been associated with.

Each request also needs to include a `<requesttime>` which is the current timestamp converted to integer.

The general structure of the URL is:

```
http://<domain>/<PrimebaseApacheHandler>/<PBAS-Name>/<Module-Name>/<Handler-Name>.htm
```

The URL to access a TeamDrive Registration Server's API is as follows:

```
https://<domain>/pbas/td2as/api/api.htm?checksum=<md5>
```

The URL to access the TeamDrive Host Server API looks as follows:

```
https://<domain>/pbas/p1_as/api/api.htm?checksum=<md5>
```

Please replace `<domain>` with the host name of the Host or Registration Server you want to connect to. `<md5>` needs to be replaced with the checksum of the current API request.

If you are accessing the API over a local network or a VPN, you can use plain HTTP. However, when sending the data over an insecure network, you must use HTTPS for security reasons.

Note: API access is verified by the IP address the request originated from. On the Registration Server, check the setting `API_IP_ACCESS` (*"Edit Provider Settings" -> "API" -> "API_IP_ACCESS"* via the Administration Console) and make sure that the external IP address of the system performing the API call is included in the list.

On the Host Server, the IP address must be added to the configuration setting `API_IP_LIST`.

14.1.1 Example API Call

The following shell script example outlines how an API call is generated and how the required MD5 checksum is calculated. In this example `curl` is used to perform the actual API call. The result is printed to the console:

```
#!/bin/sh

URL="http://regserver.yourdomain.com/pbas/td2api/api/api.htm"
CHECKSUM="<APIChecksumSalt>"
TIMESTAMP=`date "+%s"`
REQUEST="<?xml version='1.0' encoding='UTF-8' ?>\
<teamdrive><apiversion>1.0.004</apiversion>\
<command>loginuser</command>\
<requesttime>$TIMESTAMP</requesttime>\
<username>YourUserName</username>\
<password>YourPassword</password></teamdrive>"
MD5=`echo -n "$REQUEST$CHECKSUM" | md5sum | cut -f1 -d" "`
curl -d "$REQUEST" "$URL?checksum=$MD5"
```

14.1.2 API Usage Recommendations

On your side of the (web-) application, you must ensure that only successfully logged in users can view or change their own data. Users should never be allowed to view data from other TeamDrive Users. Only users associated with your provider code can be managed with API calls coming from your IP. For users with a foreign provider code you will receive a URL which must be displayed to the user so that they can login to the website of their provider.

14.1.3 Error Handling

The following errors can occur due to misconfiguration or service failures, they may not return valid XML. Your application should handle these failures appropriately.

Wrong Apache configuration

Request:

```
http://<domain>/pbas/td2api/api/service.html
```

Answer:

```
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /td2api/api/service.html was not found on this server.</p>
<hr>
<address>Apache/2.2.9 (Fedora) Server Port 80</address>
</body></html>
```

Apache handler could not be addressed

Request:

```
http://<domain>/pbas/td2api/api/service.html
```

Answer:

```
<head><TITLE>503 Service Unavailable</TITLE></head><body><H1>503 Service
Unavailable</H1><P><H3>Error Processing Request</H3></body> The following
error occurred in \'SendCGIRequest("td2api")\' while processing
your request: -12996 (-12946).
```

Application errors

Application errors will return error messages in an XML format like this:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.004</apiversion>
        <exception>
                <primarycode></primarycode>
                <secondarycode></secondarycode>
                <message></message>
        </exception>
</teamdrive>
```

`<primarycode>` and `<secondarycode>` (optional) are integer values. `<message>` is a text.

Error codes regarding the API will start at -30100 (see *Error Codes* (page 122)).

General errors with the PrimeBase Virtual Machine or database connection are in the range between 0 and -23000.

Programming errors

If a program error occurs, the server will return an error similar to the following one:

```
<HTML><HEAD><TITLE>Execution Error</TITLE></HEAD><BODY>
<H2>Execution Error</H2><FONT SIZE = +1>An error occured while processing
your request: <BR>Primary error code: <B>-10005</B>, Secondary error code:
<B>0</B><BR><FONT SIZE = 0><H3>"api_init.sys"@client line 7: ';' token
expected in place of 'execute'.</H3></BODY></HTML>
```

Invalid Requests

Invalid requests will return one of the following errors:

Unknown IP Address

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.004</apiversion>
        <exception>
                <primarycode>-30000</primarycode>
                <secondarycode></secondarycode>
                <message>Access denied</message>
        </exception>
</teamdrive>
```

Invalid Command

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.004</apiversion>
        <exception>
                <primarycode>-30001</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid Command</message>
```

```
        </exception>
</teamdrive>
```

Invalid Request

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.004</apiversion>
        <exception>
                <primarycode>-30002</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid Request</message>
        </exception>
</teamdrive>
```

Invalid XML

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.004</apiversion>
        <exception>
                <primarycode>-30003</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid XML</message>
        </exception>
</teamdrive>
```

14.2 Registration-Server API Calls

Note: Changes to API release 1.0.003:

The `<distributor>` tag was added to API release 1.0.003. The value is optional, but must be used if more than one provider exists and the admin console will be used or own API requests will be send from one IP address to access diffent provider (for details see explanation in *API Settings (<APIAccess> XML block)* (page 58)). The API setting (see *APIAllowSettingDistributor* (page 51)) must be enabled to allow setting the provider.

14.2.1 Login

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>loginuser</command>
        <requesttime></requesttime>
        <username></username>
        <password></password>
        <distributor></distributor>
</teamdrive>
```

Note: For release 1.0.003: `<useroremail>` instead of `<username>` will be accepted

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <userdata>
                <username></username>
                <email></email>
                <language></language>
                <reference></reference>
                <department></department>
        </userdata>
</teamdrive>
```

Changes to API release 1.0.003:

`<invitecode>` is not longer returned, `<reference>` and `<department>` were added.

Error Cases

User Unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30100</primarycode>
                <secondarycode></secondarycode>
                <message>Username does not exists</message>
        </exception>
</teamdrive>
```

Wrong Password

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30101</primarycode>
                <secondarycode></secondarycode>
                <message>Wrong password</message>
        </exception>
</teamdrive>
```

Account not Activated

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
```

```
                <primarycode>-30102</primarycode>
                <secondarycode></secondarycode>
                <message>Account not Activated by activation mail</message>
        </exception>
</teamdrive>
```

Account Disabled (added in 1.0.003)

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30119</primarycode>
                <secondarycode></secondarycode>
                <message>Account is disabled</message>
        </exception>
</teamdrive>
```

Account in Deletion (added in 1.0.003)

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30120</primarycode>
                <secondarycode></secondarycode>
                <message>Account will be deleted by the user</message>
        </exception>
</teamdrive>
```

Redirect to the Website of the Distributor

Users who try to login to the TeamDrive website will be redirected to the website of the distributor. The XML answer will return the URL in the `<message>`-Tag. You have to redirect the user to that URL.

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30004</primarycode>
                <secondarycode></secondarycode>
                <message>[URL]</message>
        </exception>
</teamdrive>
```

14.2.2 Search user

Warning: This function is for internal usage only. Do not allow public access.

You can perform a wildcard search using '*', e.g. 'Teamdrive*' will find all data starting with 'Teamdrive...', '*Teamdrive' will find all data ending with '...Teamdrive', and '*Teamdrive*' will find all data that contains '...Teamdrive...'. You can search by Username, Email, or both. If you search by both, the fields will be combined with an `AND`.

If you search without the wildcard, we will do an exact match for the string. A minimum of 3 characters (without wildcard) is required.

You can limit the search to your own users by using `<onlyownusers>true</onlyownusers>`. In this case you could leave username and email empty to retrieve a list with all of your users.

Currently, the answer will return a maximum of 50 records. This maximum value might change in the future. The current maximum value is included within the reply's `<maximum>`-tag.

If `<current>` == `<maximum>`, there may be more records available than returned in the reply. To retrieve the next set of records, resend the same request and put the highest User-ID from the last reply into the `<startid>` field. For the first search request you can set''<startid>'' to 0, or omit it entirely.

If a user does not belong to your distributor their email-field will be empty.

The `<devicelist>` block will only be returned if you send `<showdevice>true</showdevice>` in your request. The `<amount>` field (under `<devicelist>`) holds the number of devices that will be returned for this user. If the user has no TeamDrive client installations, this value will be 0.

In `<searchresult>` you will find statistical information about the found records:

<current> Amount of records in this reply

<total> Total amount of records

<maximum> Maximum amount of records the server will return in a reply

If no records are found, `<current>` and `<total>` will be 0. In this case, the `<userlist>` block will not be returned.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>searchuser</command>
        <requesttime></requesttime>
        <username></username>
        <email></email>
        <startid></startid>
        <showdevice>true/false</showdevice>
        <onlyownusers>true/false</onlyownusers>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <searchresult>
                <current></current>
                <maximum></maximum>
                <total></total>
        </searchresult>
        <userlist>
                <user>
                        <userid></userid>
                        <username></username>
                        <email></email>
                        <reference></reference>
                        <department></department>
```

```
                        <distributor></distributor>
                        <usercreated></usercreated>
                        <language></language>
                        <status></status>
                        <devicelist>
                                <amount></amount>
                                <device>
                                        <deviceid></deviceid>
                                        <status></status>
                                        <licensekey></licensekey>
                                        <feature></feature>
                                        <devicecreated></devicecreated>
                                        <deviceactive></deviceactive>
                                        <version></version>
                                        <platform></platform>
                                </device>
                                <device>
                                        ...
                                </device>
                        </devicelist>
                </user>
                <user>
                        ...
                </user>
        </userlist>
</teamdrive>
```

Changes to API release 1.0.003:

`<reference>` and `<department>` was added.

Error Cases

Search String too Short

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30116</primarycode>
                <secondarycode></secondarycode>
                <message>Search string to short</message>
        </exception>
</teamdrive>
```

14.2.3 Get User Data

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>getuserdata</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <userdata>
                <username></username>
                <email></email>
                <language></language>
                <distributor></distributor>
                <reference></reference>
                <department></department>
        </userdata>
        <licensedata>
                <license>
                        <created></created>
                        <productid></productid>
                        <productname></productname>
                        <type></type>
                        <number></number>
                        <featurevalue></featurevalue>
                        <featuretext></featuretext>
                        <validuntil></validuntil>
                        <limit></limit>
                        <used></used>
                        <status></status>
                        <isdefault></isdefault>
                </license>
                <license>...</license>
                <license>...</license>
        </licensedata>
        <depotdata>
                <count></count>
                <depot>
                        <hosturl></hosturl>
                        <depotid></depotid>
                        <isdefault></isdefault>
                </depot>
                <depot>...</depot>
        </ depotdata>
</teamdrive>
```

For a description of the fields, see *Get license data for a user* (page 99).

Changes to API release 1.0.003:

`<invitecode>` is not longer returned,

`<reference>` and <department> were added,

`<depotdata>`-block changed, because each user could have more than one depot, but only one default-depot. The amount of depots for the users could be found in `<count>`

A default license is normally created when the user installs their first client application. You can create a default license with the API, if no default license already exists, by setting "CreateDefaultLicense" as described in the registration server documentation.

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

14.2.4 Create a new Account

Note: The range of possible `Usernames` depend on the `RegNameComplexity` setting as described in the Registration Server documentation. Similary, the length of passwords can be restricted by the `ClientPasswordLength` setting. However, these restriction only apply when creating a new account, they will not be checked when a user attempts to log in

In case that `UseEmailAsReference` is defined, the username will be automatically generated and the email will be used to reference the user. All further request where username is was required can then also be sent with the user's email in place of their username.

The user will get an activation email sent to their email address. You can change this behaviour with the `AllowActivationWithoutEmail` setting as described in the Registration Server documentation.

The user's record will be assigned to the distributor. Users are mapped to distributors by either the IP-address of the request sender or the distributor-tag from the request. A Distributor can only create users belonging to them.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>registeruser</command>
        <requesttime></requesttime>
        <username></username>
        <useremail></useremail>
        <password></password>
        <language></language>
        <reference></reference>
        <department></department>
        <distributor></distributor>
</teamdrive>
```

Changes to API release 1.0.003:

`<reference>` and `<department>` were added

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

Username Already Exists

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30103</primarycode>
                <secondarycode></secondarycode>
                <message>Username already exists</message>
        </exception>
</teamdrive>
```

Email is already is use

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30104</primarycode>
                <secondarycode></secondarycode>
                <message>Email already exists</message>
        </exception>
</teamdrive>
```

This error will only be returned from our own TDRS because the email field is unique there.

Username Invalid, if length == 0

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30108</primarycode>
                <secondarycode></secondarycode>
                <message>Username invalid</message>
        </exception>
</teamdrive>
```

Password Invalid, if length < ClientPasswordLength

The value `ClientPasswordLength` is described in the Registration Server documentation.

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30109</primarycode>
                <secondarycode></secondarycode>
                <message>Password invalid</message>
```

```
        </exception>
</teamdrive>
```

Email Invalid, if length == 0 or missing "@" and "."

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30110</primarycode>
                <secondarycode></secondarycode>
                <message>Email invalid</message>
        </exception>
</teamdrive>
```

14.2.5 Resend Activation (added in 1.0.004)

Will resend the activation mail to the user.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>resendactivation</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30100</primarycode>
                <secondarycode></secondarycode>
                <message>Username does not exists</message>
        </exception>
</teamdrive>
```

Account already activated

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30118</primarycode>
                <secondarycode></secondarycode>
                <message>Account already activated</message>
        </exception>
</teamdrive>
```

14.2.6 Activate User

Note: To activate the user you have to send back the activation code in `<activationcode>`. This is the code that was sent to the user in the activation mail.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>activateuser</command>
        <requesttime></requesttime>
        <username></username>
        <activationcode></activationcode>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30100</primarycode>
                <secondarycode></secondarycode>
                <message>Username does not exists</message>
        </exception>
</teamdrive>
```

Wrong Activation Code

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30106</primarycode>
                <secondarycode></secondarycode>
                <message>Wrong activation code</message>
        </exception>
</teamdrive>
```

14.2.7 Activate Client (added in 1.0.003)

If a user creates their own account using a TeamDrive client application, they will be sent a *client* activation email. The activation link from that email will normally lead back to the TDRS. However, if the link does not directly point to the TDRS, the following API call can be used to activate the client.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>activateclient</command>
        <requesttime></requesttime>
        <activationcode></activationcode>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

Wrong Activation Code

See above

Activation Code not Found

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30117</primarycode>
                <secondarycode></secondarycode>
                <message>Activation code not found</message>
        </exception>
</teamdrive>
```

14.2.8 Forgotten Password

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>sendpassword</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

14.2.9 Reset Password (added in 1.0.003)

Resetting a user's password will set it to a random value. The user then had to define a new password themselves.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>resetpassword</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

14.2.10 Change password

Note: `<tmppassword>` must contain the temporary password that was emailed to the after the *sendpassword* (page 86) API call. The `<password>` filed must contain the new password chosen by the user.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>changepassword</command>
        <requesttime></requesttime>
        <username></username>
        <tmppassword></tmppassword>
        <password></password>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Temporary password wrong

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30105</primarycode>
                <secondarycode></secondarycode>
                <message>Temporary password does not match</message>
        </exception>
</teamdrive>
```

Password invalid, if length < ClientPasswordLength

The `ClientPasswordLength` value is described in the registration server documentation.

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30109</primarycode>
                <secondarycode></secondarycode>
                <message>Password invalid</message>
        </exception>
</teamdrive>
```

Temporary password expired

Note: The User has 10 minutes to complete password changing operation. If the time expired, the user has to request a new password again using *changepassword* (page 88) .

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30113</primarycode>
                <secondarycode></secondarycode>
                <message>Temporary password expired</message>
        </exception>
</teamdrive>
```

14.2.11 Update password

Note: If a user is already logged in they can change their password directly. You have to make sure that this API command is only sent for users which have already been authenticated.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>updatepassword</command>
        <requesttime></requesttime>
        <username></username>
        <newpassword></newpassword>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Password invalid, if length < ClientPasswordLength

The `ClientPasswordLength` value is described in the registration server documentation.

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30109</primarycode>
                <secondarycode></secondarycode>
                <message>Password invalid</message>
```

```
        </exception>
</teamdrive>
```

14.2.12 Set reference (added in 1.0.004)

Will set the external reference for a user

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setreference</command>
        <requesttime></requesttime>
        <username></username>
        <reference></reference>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

14.2.13 Set email (added in 1.0.003)

Note: This command will change the email for the user directly without sending a confirmation email to the user like the command *changeemail* (page 92).

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setemail</command>
        <requesttime></requesttime>
        <username></username>
        <newemail></newemail>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

Email invalid, if length == 0 or "@" and "." are missing

See above

14.2.14 Change email address

Note: The change-email request will be stored until the user confirms the new email address. Until then, the old email will still be displayed.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>changeemail</command>
        <requesttime></requesttime>
        <username></username>
        <newemail></newemail>
```

```
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Email already exists (in another user's registration or in their own registration)

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30104</primarycode>
                <secondarycode></secondarycode>
                <message>Email already exists</message>
        </exception>
</teamdrive>
```

This error will only be returned from our own TDRS, because the email field is unique there.

Email invalid, if length == 0 or "@" and "." are missing

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <
                primarycode>-30110</primarycode>
```

```
                <secondarycode></secondarycode>
                <message>Email invalid</message>
        </exception>
</teamdrive>
```

14.2.15 Confirm new email

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>confirmnewemail</command>
        <requesttime></requesttime>
        <username></username>
        <activationcode></activationcode>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Wrong activation code

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30106</primarycode>
                <secondarycode></secondarycode>
```

```
                <message>Wrong activation code</message>
        </exception>
</teamdrive>
```

Email already exists in an other registration

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30104</primarycode>
                <secondarycode></secondarycode>
                <message>Email already exists</message>
        </exception>
</teamdrive>
```

This error will only be returned from our own TDRS, because the email field is unique there.

14.2.16 Change language

Note: Languages fields use valid ISO 3166 language codes (see http://en.wikipedia.org/wiki/ISO_3166-1).

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>changelanguage</command>
        <requesttime></requesttime>
        <username></username>
        <newlanguage></newlanguage>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Invalid language

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30115</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid language</message>
        </exception>
</teamdrive>
```

14.2.17 Remove user (added in 1.0.003)

This call will delete the user account immediately (as opposed to *deleteuser* (page 97) which requires user confirmation). Set `<deletelicense>` to `$true` if you would like to delete the user's license as well.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removeuser</command>
        <requesttime></requesttime>
        <username></username>
        <deletelicense>$true/$false</deletelicense>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

14.2.18 Remove device (added in 1.0.004)

This call deletes a user's device. The id of the device must specified in the request. The list of devices a user posses can be retrieved using *searchuser* (page 78).

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removedevice</command>
        <requesttime></requesttime>
        <username></username>
        <deviceid></deviceid>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Device not found

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30121</primarycode>
                <secondarycode></secondarycode>
                <message>Device not found</message>
        </exception>
</teamdrive>
```

14.2.19 Delete user

Note: The user will be send a confirmation email. Their account will not be deleted until they confirm the operation. (see *Remove user (added in 1.0.003)* (page 96) if you wish to delete a user without confirmation)

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>deleteuser</command>
        <requesttime></requesttime>
        <username></username>
```

```
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

14.2.20 Confirm delete user

Note: This function is not currently activated.

Note: The <activationcode> will be send to the email address of the user The User has to authenticate again with their password. Set `<deletedepot>` to `$true` to also delete the user's default depot. Set `<deletelicense>` to `$true` to also delete the user's licenses.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>confirmuserdelete</command>
        <requesttime></requesttime>
        <username></username>
        <password></password>
        <activationcode></activationcode>
        <deletedepot></deletedepot>
        <deletelicense></deletelicense>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Password invalid

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30101</primarycode>
                <secondarycode></secondarycode>
                <message>Wrong password</message>
        </exception>
</teamdrive>
```

Activation code invalid

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30106</primarycode>
                <secondarycode></secondarycode>
                <message>Wrong activation code</message>
        </exception>
</teamdrive>
```

14.2.21 Get license data for a user

Note: The `<licensedata>` block is identical to the reply from a login.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>getlicensedata</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <licensedata>
                <license>
                        <created></created>
                        <productid></productid>
                        <productname></productname>
                        <type></type>
                        <number></number>
                        <featurevalue></featurevalue>
                        <featuretext></featuretext>
                        <validuntil></validuntil>
                        <limit></limit>
                        <used></used>
```

```
                        <status></status>
                        <isdefault></isdefault>
                </license>
                <license>...</license>
                <license>...</license>
        </licensedata>
</teamdrive>
```

Description of the fields and values:

- `created`: Creation date, format DD.MM.YYYY
- `productid`: 1 or 2 (see productname)
- `productname`: client (1) or server (2)
- `type`: 0 = permanent, 1 = monthly payment, 2 = not for resale, 3 = yearly payment
- `number`: license number
- `featurevalue`: sum of the numbers as described in featuretext
- `featuretext`: Banner (1), WebDAVs (2), Personal (4), Professional (8), Enterprise (16)
- `validuntil`: license valid until, Format DD.MM.YYYY
- `limit`: License amount
- `used`: Used licenses
- `status`: 0 = active, 1 = deactivated, 2 = deleted (records with status "deleted" will not be returned using the API)
- `isdefault`: Is it a default license for the user (only possible for productid = 1). The user will be downgraded to their default license if a foreign license will be rejected from their installation.

Changes to API release 1.0.003:

> A default license will normally be created when the user installs their first client application. The API could create a default license, if no default license exists, by setting "CreateDefaultLicense" as described in the registration server documentation.

Error Cases

User Unknown

See above

License unknown, deactivated or deleted

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30201</primarycode>
                <secondarycode></secondarycode>
                <message>Unknown License</message>
        </exception>
</teamdrive>
```

14.2.22 Get default-license for a user

Note: The reply will return the same `<licensedata>`-Block as the command *getlicensedata* (page 99).

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>getdefaultlicense</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

see Reply in *getlicensedata* (page 99)

14.2.23 Get default depot URL by username

Note: The reply return the same `<depotdata>` block as described for the `"login"` command.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>getdefaultdepotdata</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <depotdata>
                <depot>
                        <depotid></depotid>
                        <name></name>
                        <status></status>
                        <accountnumber></accountnumber>
                        <created></created>
                        <storagelimit></storagelimit>
                        <storageused></storageused>
                        <transferlimit></transferlimit>
                        <transferused></transferused>
                        <userlist></userlist>
                </depot>
                <depot>...</depot>
                <depot>...</depot>
        </depotdata>
</teamdrive>
```

Error Cases

User Unknown

See above

Depot unknown

Note: Users without a TeamDrive Client installation might have no depot.

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30107</primarycode>
                <secondarycode></secondarycode>
                <message>No Default Depot</message>
        </exception>
</teamdrive>
```

14.2.24 Get url for the default depot server (added in 1.0.004)

Note: This call could be used to get the current default depot server which is selected in the admin console to create depots on the host server using the API

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>gethostfordepot</command>
        <requesttime></requesttime>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <hosturl></hosturl>
</teamdrive>
```

Error Cases

No default depot server

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30107</primarycode>
                <secondarycode></secondarycode>
                <message>No default depot server</message>
        </exception>
</teamdrive>
```

14.2.25 Set depot for user (added in 1.0.003)

`<sendtoclient>` is an optional parameter and will do the `sendinvitation` call as described below (added in 1.0.004)

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setdepotforuser</command>
        <requesttime></requesttime>
        <username></username>
        <depot></depot>
        <isdefault></isdefault>
        <sendtoclient></sendtoclient>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

Depot already exists

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30307</primarycode>
                <secondarycode></secondarycode>
                <message>Depot already exists</message>
        </exception>
</teamdrive>
```

14.2.26 Remove depot from user (added in 1.0.005)

Note: In case of deleting the default depot of a user and the user has still an other depot in his list, the oldest depot will become the default depot

`<sendtoclient>` is an optional parameter and will do the `sendinvitation` call as described below

There are different calls possible:

- Sending the depot file like in `setdepotforuser`
- Sending HostURL and depot id

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removedepotfromuser</command>
        <requesttime></requesttime>
        <username></username>
        <depot></depot>
        <sendtoclient></sendtoclient>
</teamdrive>
```

OR

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removedepotfromuser</command>
        <requesttime></requesttime>
        <username></username>
        <hosturl></hosturl>
        <depotid></depotid>
        <sendtoclient></sendtoclient>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

Depot data missing or invalid

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30123</primarycode>
                <secondarycode></secondarycode>
                <message>Depot data missing or invalid</message>
        </exception>
</teamdrive>
```

Depot not found

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30124</primarycode>
                <secondarycode></secondarycode>
                <message>Depot not found</message>
        </exception>
</teamdrive>
```

14.2.27 Send invitation

Note: Will be used for business users to distribute or delete a company depot to other users.

`<type>` is `INV_TYPE_CREATEDEPOT` or `INV_TYPE_DELETEDEPOT`

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>sendinvitation</command>
        <requesttime></requesttime>
        <username></username>
        <userlist></userlist>
        <type></type>
        <invitation></invitation>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Type unknown

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30111</primarycode>
                <secondarycode></secondarycode>
                <message>Invitation type unknown</message>
        </exception>
</teamdrive>
```

14.2.28 Set invited user

Note:

Will be used for the referral program. (see *REFERRAL Settings* (page 64)).

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setinviteduser</command>
        <requesttime></requesttime>
        <username></username>
        <inviteduser></inviteduser>
        <sendmail></sendmail>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Username invalid

The invited user could not be found

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30108</primarycode>
                <secondarycode></secondarycode>
                <message>Unknown user: xxx</message>
        </exception>
</teamdrive>
```

Set invited user failed

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30209</primarycode>
                <secondarycode></secondarycode>
                <message>Setinvited user failed</message>
        </exception>
</teamdrive>
```

14.2.29 Create license

Parameters:

- `<productname>` : server, client
- `<type>` : monthly, yearly, permanent
- `<featurevalue>` : One of the following values: webdavs, personal, professional, enterprise, banner (only for client licenses)
- `<limit>` : Amount (for a client license), 0000 = unlimited (for a server license)

  Added with version 1.0.004:
- `<licensereference>`: An optional external reference (free text field with 100 char)
- `<contractnumber>`: An optional contract number (free text field with 255 char)
- `<validuntil>`: An optional valid-until date. Format must be DD.MM.YYYY
- `<changeid>`: An optional change id for license changes
- `<sendemail>`: Currently not used

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>createlicense</command>
        <requesttime></requesttime>
        <username></username>
        <productname></productname>
        <type></type>
        <featurevalue></featurevalue>
        <limit></limit>
        <licensereference></licensereference>
        <contractnumber></contractnumber>
        <validuntil></validuntil>
        <changeid></changeid>
        <sendemail></sendemail>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

Productname unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30203</primarycode>
                <secondarycode></secondarycode>
                <message>Productname unknown</message>
```

```
        </exception>
</teamdrive>
```

Type unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30204</primarycode>
                <secondarycode></secondarycode>
                <message>Type unknown</message>
        </exception>
</teamdrive>
```

Feature unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30205</primarycode>
                <secondarycode></secondarycode>
                <message>Feature unknown</message>
        </exception>
</teamdrive>
```

Limit unknown / invalid

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30206</primarycode>
                <secondarycode></secondarycode>
                <message>Limit unknown</message>
        </exception>
</teamdrive>
```

Invalid date (added in 1.0.004)

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30122</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid date</message>
        </exception>
</teamdrive>
```

14.2.30 Create license without user (added in 1.0.003)

Similar to *createlicense* (page 107), but without setting a reference to a specific user. Might be useful if a shop application wanted to retrieve a license before the user has an account in the shop.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>createlicensewithoutuser</command>
        <requesttime></requesttime>
        <username></username>
        <productname></productname>
        <type></type>
        <featurevalue></featurevalue>
        <limit></limit>
        <licensereference></licensereference>
        <contractnumber></contractnumber>
        <validuntil></validuntil>
        <changeid></changeid>
        <sendemail></sendemail>
        <distributor></distributor>
</teamdrive>
```

Reply and errors identical to *createlicense* (page 107).

14.2.31 Assign user to license (added in 1.0.003)

If *createlicensewithoutuser* (page 110) was used then this command can be used to assign the license to a user. The user's record information will be used to update the values in the license table.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>assignusertolicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

License unknown

See above

14.2.32 Assign license to client (added in 1.0.004)

Assigns a license to a teamdrive client. `<devicelist>` is an optional list of devices the user posses. If empty, all of the user's devices will be used. This function can only be used if the user has at least one device.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>assignlicensetoclient</command>
        <requesttime></requesttime>
        <username></username>
        <devicelist></devicelist>
        <number></number>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled

See above

Account in Deletion

See above

License unknown

See above

Device not found

See above

14.2.33 Remove user from license (added in 1.0.003)

The complement to *assignusertolicense* (page 110) which can remove a user's license.

Important: This will not remove the license from any installations!

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removeuserfromlicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

Account Disabled (added in 1.0.003)

See above

Account in Deletion (added in 1.0.003)

See above

License unknown

See above

Invalid distributor

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30114</primarycode>
                <secondarycode></secondarycode>
                <message>Invalid Distributor</message>
        </exception>
</teamdrive>
```

14.2.34 Deactivate license (added in 1.0.003)

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>deactivatelicense</command>
        <requesttime></requesttime>
        <number></number>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

License unknown

See above

Invalid distributor

See above

License change failed

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30210</primarycode>
                <secondarycode></secondarycode>
                <message>License already disabled</message>
        </exception>
</teamdrive>
```

14.2.35 Activate license (added in 1.0.003)

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>activatelicense</command>
        <requesttime></requesttime>
        <number></number>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

License unknown

See above

Invalid distributor

See above

License change failed

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30210</primarycode>
                <secondarycode></secondarycode>
                <message>License not disabled</message>
        </exception>
</teamdrive>
```

14.2.36 Upgrade license

Fixed in version 1.0.004: The current feature was ignored.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>upgradelicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <featurevalue></featurevalue>
        <limit></limit>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

License unknown

See above

License upgrade not possible

Note: This error occurs in cases such as when the license is deactivated or deleted.

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30202</primarycode>
                <secondarycode></secondarycode>
                <message>License upgrade failed</message>
        </exception>
</teamdrive>
```

14.2.37 Upgrade default-license

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>upgradedefaultlicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <featurevalue></featurevalue>
        <limit></limit>
        <distributor></distributor>
</teamdrive>
```

Reply:

For form of reply and possible errors see *upgradelicense* (page 115).

14.2.38 Downgrade license

Fixed in version 1.0.004: Downgrading a license was not working

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>downgradelicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <featurevalue></featurevalue>
        <decreaselimit></decreaselimit>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

License unknown

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30201</primarycode>
                <secondarycode></secondarycode>
                <message>Unknown License</message>
        </exception>
</teamdrive>
```

Downgrade not possible

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30208</primarycode>
                <secondarycode></secondarycode>
                <message>License downgrade failed</message>
        </exception>
</teamdrive>
```

14.2.39 Downgrade default-license

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>downgradedefaultlicense</command>
        <requesttime></requesttime>
        <username></username>
        <number></number>
        <featurevalue></featurevalue>
        <limit></limit>
        <distributor></distributor>
</teamdrive>
```

Reply:

For reply and errors see *downgradelicense* (page 116).

14.2.40 Show used client licenses

Note: `<licensenumber>` is optional.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>getusedlicense</command>
        <requesttime></requesttime>
        <username></username>
        <licensenumber></licensenumber>
        <distributor></distributor>
</teamdrive>
```

Reply:

Note: `<userlist>` is a comma separated list with user names. Only active license will be returned (not inactive or deleted licenses). If no license data could be found, error-30201 will be returned (see below).

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <licensedata>
                <license>
                        <number></number>
                        <used></used>
                        <limit></limit>
                        <userlist></userlist>
                </license>
                <license>...</license>
                <license>...</license>
        </licensedata>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

License unknown, deactivated or deleted

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30201</primarycode>
                <secondarycode></secondarycode>
                <message>Unknown License</message>
```

```
        </exception>
</teamdrive>
```

14.2.41 Set license reference (added in 1.0.004)

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setlicensereference</command>
        <requesttime></requesttime>
        <licensenumber></licensenumber>
        <licensereference></licensereference>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

License unknown, deactivated or deleted

See above

14.2.42 Remove license from a user

Note: `<userlist>` is a comma separated list with user names.

Added in version 1.0.004: `<devicelist>` is an optional list of devices belonging to the user. If it is empty or unspecified, all of the user's devices will be used.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>removelicense</command>
        <requesttime></requesttime>
        <username></username>
        <licensenumber></licensenumber>
        <userlist></userlist>
        <devicelist></devicelist>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

License unknown, deactivated or deleted

See above

14.2.43 Delete / cancel license

Note: If no license data could be found, you will get an "Unknown License" error. Use the field `<decreaselimit>` to define the amount of licenses. To delete / cancel the license completely, send 0 in the field, otherwise a value > 0.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>cancellicense</command>
        <requesttime></requesttime>
        <username></username>
        <licensenumber></licensenumber>
        <decreaselimit></decreaselimit>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

User Unknown

See above

Account not Activated

See above

License could not be reduced, because they are still in use

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30207</primarycode>
                <secondarycode></secondarycode>
                <message>Cancel license failed</message>
        </exception>
</teamdrive>
```

License unknown, deactivated or deleted

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <exception>
                <primarycode>-30201</primarycode>
                <secondarycode></secondarycode>
                <message>Unknown License</message>
        </exception>
</teamdrive>
```

14.2.44 Set distributor

Note: This function can currently only be accessed by the default distributor. Only registered distributors can be used.

Request:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <command>setdistributor</command>
        <requesttime></requesttime>
        <username></username>
        <distributor></distributor>
</teamdrive>
```

Reply:

```
<?xml version='1.0' encoding='UTF-8' ?>
<teamdrive>
        <apiversion>1.0.005</apiversion>
        <intresult>0</intresult>
</teamdrive>
```

Error Cases

Access denied

See above

Invalid Distributor

See above

14.3 Error Codes

The following table lists all API-Error-Codes that might be returned.

Registration-Server-Error-Codes:

Table 14.1: API Error Codes

Primary	Message	Comment
-30000	Access denied	
-30001	Invalid Command	
-30002	Invalid Request	
-30003	Invalid XML	
-30004	URL	This user will be handled using the webinterface of the distributor
-30005	Maintenance work	A 503 from the API-Server should be displayed as Maintenance work for the user. 503 will be mapped to -30005.
-30100	Username does not exist	
-30101	Wrong password	
-30102	Account not activated by activation mail	
-30103	Username already exists	
-30104	Email already exists	No longer used in API 1.0.003
-30105	Temporary password does not match	
-30106	Wrong activation code	
-30107	No Default Depot	
-30108	Username invalid	
-30109	Password invalid	
-30110	Email invalid	
-30111	Invitation type unknown	
-30112	Invalid location	
-30113	Temporary password expired	
-30114	Distributor of the user does not match in the database	
-30115	Invalid language	Currently not in use
-30116	Search string to short	
-30117	Activation code not found	
-30118	Account already activated	Currently not in use
-30119	Account disabled	
-30120	Account will be deleted	
-30121	Device not found	
-30122	Invalid date	
-30201	Unknown License	
-30202	License Upgrade failed	
-30203	Productname unknown	
-30204	Type unknown	
-30205	Feature unknown	
-30206	Limit unknown	
-30207	Cancel license failed	
-30208	Downgrade license failed	
-30209	Empty list	Currently not in use
-30210	License change failed	

Continued on next page

Table 14.1 – continued from previous page

Primary	Message	Comment
-30211	License in use	Currently not in use
-30301	No Depot for User	
-30302	Depot-ID does not match	
-30303	Space-ID does not match	
-30304	Increasing Depot failed	
-30305	Decreasing Depot failed	
-30306	Invalid storage limit	
-30307	Depot already exists	

C

Client, 1

D

Device, 1

I

Installation, 1

P

Provider (aka Distributor or Tenant), 1

S

Space, 1

U

User, 1