EPSON

Fiscal Germany

TSE Developer's Guide

Revision 1.0.1

TABLE OF CONTENTS

1 Purpose of this Document

This document describes the technical integration of the EPSON TSE hosted on the printer, PC or EPS TSE Server and gives information on how to control it. This document does **not** contain any information on the requirements of the data contents logged to the TSE. For this kind of information, please consult the information provided by BMF and BSI. You can find online links in the chapter Official Government Legislation Documents at the end of this document.

If you require additional information about the printer, PC or EPS TSE Server itself or on its configuration, please refer to the documentation that comes with the device or contact your sales representative.

2 Development TSEs

2.1 What is a Development TSE and how can it be identified?

A Development TSE is a device that is nearly completely like a normal TSE, with those differences:

- It is not a certified product and may not be used in productive POS system.
- It is possible to perform a factory reset on the Development TSE.
- Development TSEs have a shorter certificate validity as certified TSEs.

If there is a confusion between certified and Development TSEs, it can be difficult to tell them apart again. From the outside, there is no visible difference. If you perform GetStorageInfo on the TSE, you can tell it by the certificate expiration date. To be sure, you can try to perform the function FactoryReset. If it succeeds, it is a Development TSE, on a certified TSE the function will fail.

2.2 Signing

2.2.1 Certificate Validity

The validity of the certificate on the Development TSEs is shorter than the validity of the certificate on the final product.

2.2.2 No BSI Certification

The Development TSEs are not certified TSEs for use in a production POS! When used in a production environment, the owner will be liable for not using a certified TSE.

2.3 API

2.3.1 FactoryReset

If you use the function FactoryReset to reset the Development TSE, you need to reboot the printer afterwards. In case of the EPS TSE Server or the Windows driver, please power cycle the TSE by removing and reinserting it.

Please note that this function exists only for development purposes and is not available for the production TSEs.

3 How to Start Development

3.1 Install the SDK of Your Choice

Select the matching SDK for the platform of your application. If needed, install it to your development environment as described in the accompanying manual (see Related Documents below).

The following table gives you an overview which SDK can be used for which platform:

SDK	Platform(s)	Connection	Host	Installation needed? (in development environment)
ePOS SDK for Android	Android	WLAN, Bluetooth, USB to printer	Printer, Server	Yes
ePOS SDK for iOS	iOS	WLAN, Bluetooth, USB to printer	Printer, Server	Yes
ePOS SDK for JavaScript	Browser	Ethernet, WLAN	Printer	Yes
ePOS Device XML	Windows, Linux, MacOS, Android, iOS	Ethernet, WLAN, local USB (via Windows Driver)	Printer, Server, PC	No
ESC/POS	Windows, Linux, MacOS, Android, iOS	Ethernet, WLAN, USB to printer (via Virtual Port Driver), Bluetooth	Printer	No
Windows SDK	Windows	Ethernet, WLAN, USB to printer (via Virtual Port Driver), Bluetooth, local USB (via Windows Driver)	PC	Yes

Please note that ePOS Device XML is the protocol that works in all configurations. If you plan to support different architectures (some TSE in the printer, some on the server and / or some locally connected) then it is advisable to use ePOS Device XML, as it has to be implemented only once for all architectures. Also, it is complete platform agnostic on the client side: Anything that can open a socket is supported.

3.2 Get Familiar With the SDK

If you are not familiar yet with ePOS, use the SDK manuals to get your way around in the chosen SDK.

3.3 Connect to the TSE Device

Get the TSE-Host up and running with the TSE connected. Please find more information about that in the chapter Host Configuration. Use the connect method of the respective SDK to connect to the TSE. Check the SDK manuals for details on how to do that. Below there will be a description using ePOS Device XML.

3.4 EpsonTSEDemo.exe

Another good starting point is the EpsonTSEDemo.exe. This is a demo program for Windows that implements all functions of the TSE. The tool as well as the full source code in c++ are available on epson-biz.com (See Epson Online Resources).

Please refer to this tool:

- For reference source code
- Understanding functions by role (Admin, TimeAdmin)
- Identifying issues with Epson firmware or drivers vs. your code

4 Host Configuration

The TSE-Host is either a printer (TM-m30/II series or TM-T88VI-iHub), the EPS TSE Server 3/8 or a Windows PC using the Windows Driver. If in this guide the term TSE-Host is mentioned it refers to either one of printer, PC or server.

4.1 Configure the Printer

To configure the printer please use the Epson TM Utility for your printer model or connect to the printer with a browser to access Webconfig. Please refer to the printer's manual for details.

4.2 Configure the Server

To configure the EPS TSE Server, please use the utility provided with the EPS TSE Server. The utility is available for Mac and Windows.

Connect the EPS TSE Server to the local network. Install and start the utility. It will automatically scan for available servers. Even if the EPS TSE Server is not configured yet, it will be discovered and show up in the server list. Use the utility to configure the EPS TSE Server for your network. For details please refer to the help function, and the documentation provided with the EPS TSE Server.

4.3 Install the Windows Driver

The Epson TSE Driver allows the connection of a TSE directly to a Windows PC and comes in 5 different packages. The differences are as follows:

EpsonTSEDriver_x.exe:	This package contains all versions of the Epson TSE Driver for XP and later versions of Windows in 32 and 64 Bit. It also contains the .NET 4.0 runtime which is needed by the Epson TSE Driver.
EpsonTSEDriver_x_32Bit.exe:	This package contains only the Epson TSE Driver for Windows XP 32 Bit. The .NET 4.0 runtime must be installed separately beforehand.
EpsonTSEDriver_x_32BitXP.exe:	This package contains only the Epson TSE Driver for Windows 7 or higher 32 Bit. The .NET 4.0 runtime must be installed separately beforehand.
EpsonTSEDriver_x_64Bit.exe:	This package contains only the Epson TSE Driver for Windows XP 64 Bit. The .NET 4.0 runtime must be installed separately beforehand.
EpsonTSEDriver_x_64BitXP.exe:	This package contains only the Epson TSE Driver for Windows 7 or higher 64 Bit. The .NET 4.0 runtime must be installed separately beforehand.

5 Accessing the TSE

The TSE can be accessed via the ePOS Device XML protocol, the iOS SDK, the Android SDK, the JavaScript SDK and ESC/POS. The ePOS Device XML protocol is described in detail, for the iOS and Android SDK there is just an overview – please refer to the respective SDK manuals for more information.

Please note that not every TSE host supports every connection method. Please consult the following table:

Host	SDK/Protocol
Printer	ePOS Device XML iOS SDK (using ESC/POS or XML) Android SDK (using ESC/POS or XML) JavaScript SDK (using JSON) Windows SDK (using ESC/POS or XML)
EPS TSE Server 3/8	ePOS Device XML iOS SDK (using XML) Android SDK (using XML) Windows SDK (using XML)
Windows Driver	ePOS Device XML Windows SDK (using XML)

5.1 ePOS Device XML

5.1.1 Connecting

To connect the application to the TSE-Host, TCP-Sockets are used. Just open a socket to the IP-address of the TSE-Host to the port 8009 (unencrypted) or 8143 (encrypted/SSL). If you use SSL, you need to use the appropriate client-side libraries for SSL encryption.

If the connection is successful, the TSE-Host will immediately send a `connect` message which can be read from the socket by the application:

```
<connect>
  <data>
    <client_id>sock3514555410</client_id>
    <protocol_version>2</protocol_version>
  </data>
</connect>"\0"
```

Please not the trailing null character (‘\0’) in the message. All messages in ePOS Device XML must be terminated by a null character.

5.1.2 Opening a TSE

To open a TSE, send the `open` message via the socket to the TSE-Host:

```
<open_device>
  <device_id>local_TSE</device_id>
  <data>
    <type>type_storage</type>
  </data>
</open_device>"\0"
```

Don’t forget to add the null character at the end of the message and make sure the right `device_id` is inside the `<device_id>` tag.

If everything is correct, the TSE-Host will reply with the following message:

```
<open_device>
  <device_id>local_TSE</device_id>
  <code>OK</code>
  <data_id>1</data_id>
</open_device>"\0"
```

The TSE is now opened exclusively to the client. In case of concurrent access to one TSE from different clients, make sure you always open a TSE for an operation and close it again afterwards. The status of login, transactions etc. will not be affected by closing the TSE.

5.1.3 Closing the TSE

To close a TSE, send the `close` message via the socket to TSE-Host:

```
<close_device>
  <device_id>local_TSE</device_id>
</close_device>"\0"
```

The response looks like this:

```
<close_device>
  <device_id>local_TSE</device_id>
  <code>OK</code>
  <data_id>2</data_id>
</close_device>"\0"
```

All the status for the TSE having been set during operation will be kept.

5.1.4 Reconnection in case of Network Interruption

If the socket connection breaks for some reason, any opened TSE will be blocked from opening for 1 minute. The client that has opened the TSE can reconnect immediately. To do this you need the value of the last `<data_id>` tag, which comes in any reply to an ePOS Device XML message to a specific device. You also need the `client_id` provided to you in the `connect` message.

Then you open the socket to the TSE-Host again and you will receive a new connect message containing a new client id.

Now you send a reconnect message, containing the `client_id` from the old connection as `<old_client_id>`, the `client_id` from the new connection as `<new_client_id>` and the last `data_id` as `<received_id>` from the old connection.

```
<reconnect>
  <data>
    <old_client_id>sock3514555410</old_client_id>
    <new_client_id>sock3514555411</new_client_id>
    <received_id>2</received_id>
</data>
</reconnect>"\0"
```

The response will simply look like this:

```
<reconnect>
  <code>OK</code>
</reconnect>"\0"
```

If there was a pending message from the TSE-Host that could not be sent because of the disconnection, it will be sent again immediately after the `reconnect` message.

For more details and examples, please consult the ePOS Device XML manual.

5.1.5 Operating the TSE

The API of the TSE is a JSON message format. An introduction to this can be found in the next part: “Controlling the TSE”. In order to send a JSON message to the TSE via ePOS Device XML, you need to send a `device_data` message to the TSE-Host.

The `device_data` message contains the `device_id` of the TSE, which must be open, and a `data` section containing the message `type`, which is always "`operate`" in case of the TSE, a `timeout` in milliseconds, and the `requestdata`, which is the actual JSON message.

```
<device_data>
  <device_id>local_TSE</device_id>
  <data>
    <type>operate</type>
    <timeout>10000</timeout>
    <requestdata>
    [JSON request]
    </requestdata>
  </data>
</device_data>"\0"
```

The response is a `data` message:

```
<data>
<type>onoperateresult</type>
<resultdata>
[JSON response]
</resultdata>
</data>"\0"
```

5.2 iOS-SDK

5.2.1 Connecting

Create an instance of the Epos2GermanyFiscalElement class:

```
Epos2GermanyFiscalElement.init()
```

Then connect to the TSE-Host via IP address or Bluetooth Device address.

```
Epos2GermanyFiscalElement.connect(address)
```

Now you can check if a TSE is connected to the TSE-Host:

```
bool Epos2GermanyFiscalElement.getStatus()
```

Finally set the delegate to receive responses from the TSE:

```
Epos2GermanyFiscalElement.setReceiveEventDelegate(
id<Epos2GermanyFiscalElementReceiveDelegate>)
```

5.2.2 Disconnecting

To disconnect, use the following method:

```
Epos2GermanyFiscalElement.disconnect()
```

5.2.3 Reconnection in case of Connection Interruption

If the connection breaks for some reason, any opened TSE will be blocked from opening for 1 minute. The client that has opened the TSE can reconnect immediately. The reconnection is handled automatically by the SDK.

5.2.4 Operating the TSE

The API of the TSE is a JSON message format. An introduction to this can be found in the next part: “Controlling the TSE”. In order to send a JSON message to the TSE via the iOS SDK, you need to call the operate method:

```
int Epos2GermanyFiscalElement.operate(jsonString, [timeout])
```

The result of the operation will be received in JSON format by the ReceiveEventDelegate.

5.3 Android-SDK

5.3.1 Connecting

Create an instance of the Epos2GermanyFiscalElement class:

```
new Epos2GermanyFiscalElement(Context context)
```

Then connect to the TSE-Host via IP address or Bluetooth Device address.

```
void Epos2GermanyFiscalElement.connect(target)
```

Now you can check if a TSE is connected to the TSE-Host:

```
bool Epos2GermanyFiscalElement.getStatus()
```

Finally set the listener to receive responses from the TSE:

```
void Epos2GermanyFiscalElement.setReceiveEventListener(
ReceiveListener listener)
```

5.3.2 Disconnecting

To disconnect, use the following method:

```
Epos2GermanyFiscalElement.disconnect()
```

5.3.3 Reconnection in case of Connection Interruption

If the connection breaks for some reason, any opened TSE will be blocked from opening for 1 minute. The client that has opened the TSE can reconnect immediately. The reconnection is handled automatically by the SDK.

5.3.4 Operating the TSE

The API of the TSE is a JSON message format. An introduction to this can be found in the next part: “Controlling the TSE”. In order to send a JSON message to the TSE via the Android SDK, you need to call the operate method:

```
int Epos2GermanyFiscalElement.operate(jsonString, [timeout])
```

The result of the operation will be received in JSON format by the ReceiveEventDelegate.

6 JSON Messages

6.1 Request

A JSON request message has the following structure:

```
{
  "storage": {
    "type": "[storage type]",
    "vendor": "[vendor name]"
  },
  "function": "[requested function]",
  "input": {
    "Item1": "[input parameter1]",
    "Item2": "[input parameter2]",
    ......
  },
  "compress": {
    "required": true/false,
    "type": "compression type"
  }
}
```

6.1.1 Storage

The storage object has the fields `type` and `vendor`. When calling a generic command (currently only `GetStorageInfo`), please use “COMMON” for `type` and an empty string for `vendor`.

When calling a specific TSE command, please use the `type` “`TSE`”. For `vendor`, please use the value of the field `vendorType` acquired by `GetStorageInfo`, which is “`TSE1`” for the current TSE model.

6.1.2 Function/Input

The field `function` contains the desired function to be called.

The `input` object contains the parameters for the function. If the function does not require parameters, please provide an empty object.

6.1.3 Compress

The `compress` object allows to require the JSON responses of the TSE-Host to be compressed. This can be of service for very slow network connections but is usually not needed. You need to decompress the JSON responses by yourself, if you choose to active compression.

The field `required` may either be `true` or `false`. If it is `true`, the JSON responses will be compressed.

The field `type` specifies the compression type. If `required` is `false`, please provide an empty string. If `required` is `true`, only the value “`zip_deflate`” is supported. To decompress the message, you need a function implementing the decompression part of the zip-deflate algorithm.

6.1.4 Example

```
{
  "storage": {
    "type": "TSE",
    "vendor":"TSE1"
  },
  "function": "StartTransaction",
  "input": {
    "clientId": "POS1",
    "processData": "YXBwbGXjgYLjgYTjgYbjgYjjgYo=",
    "processType": "Start",
    "additionalData": ""
  },
  "compress": {
    "required": false,
    "type": ""
  }
}
```

This is only a sample transaction. Please refer to the official information from the German government about the contents of transactions in real-world applications.

6.2 Response

6.2.1 Successful Execution

```
{
  "result": "EXECUTION_OK",
  "function": "[requested function]",
  "output": {
    "Item1": "[output parameter1]",
    "Item2": "[output parameter2]"
  }
}
```

When the operation was successful, the field `result` will have the value "`EXECUTION_OK`". The field `function` contains the called function for your reference and the `output` object contains the output parameters of the function. If the function has no output, it will be an empty object.

6.2.2 Failed Execution

```
{
  "result": "[ErrorCode]",
  "function": "[requested function]",
  "error": {
    "fact": "[fact of error]"
    "errorinfo": "[error info]",
  }
}
```

When the operation failed, the field `result` will contain an error code. The field `function` contains the called function for your reference and the `error` object contains information about the error. The field `fact` will have a short description of the error, the field `errorinfo` information about the reason, if applicable.

7 Controlling the TSE

The TSE is controlled by commands in JSON format, which are being sent through the "`operate`" function in the respective SDK. For the specification please consult the document "JSON specification_um_en.pdf" in the SDK package.

7.1 User Roles

The TSE has two user roles: Admin and TimeAdmin. There is a specific PIN for each role.

7.1.1 Admin

When logged in with the Admin role, you can perform administrative tasks on the TSE like changing PINs, setting timeouts, changing the shared secret, registering clients and so on. In the JSON command specifications you can look up if a command needs the Admin role logged in.

7.1.2 TimeAdmin

The TimeAdmin role can set the time of the TSE. The TimeAdmin login stays permanent for client during operation to enable the automatic update of TSEs time by the TSE-Host.

7.2 Information

Please use the command `GetStorageInfo` to retrieve information about a TSE. The detailed description of the result can be found in the document "JSON specification_um_en.pdf" which is part of all SDKs.

The application should regularly monitor the TSE information. It contains important information about the health status (e.g. flash health) and the remaining lifetime of the TSE (e.g. signature count, certificate expiration).

Example:

```
"function": "GetStorageInfo",
"input": {},
```

7.3 Initialization

7.3.1 Setup the TSE

Call the JSON command `SetUp`. By this you will set up the PUK and the initial PINs for Admin and TimeAdmin.

Example:

```
"function": "SetUp",
"input": {
  "puk": "123456",
  "adminPIN": "12345",
  "timeAdminPIN": "54321"
},
```

Important: If SetUp fails for any reason, please use the same PINs and PUK when retrying. Otherwise the TSE can be rendered unusable.

Setup adds also a client ID for the TSE-Host to the TSE. This always starts with “EPSON_”, followed by either the serial number of the printer or TSE Server, or the host name in case of the Windows driver. Please do not use this client ID for any operation, always use the ones that identify your tills.

7.3.2 Login as Administrator to the TSE

In addition to the PINs, the login process is secured by a simple challenge-response mechanism. First you need to request a challenge string from the TSE-Host:

Example:

```
"function": "GetChallenge",
"input": {
  "userId": "Administrator"
},
```

The response object has a field "`challenge`", which contains a 16 character long random string. Now concatenate the string with the shared secret, which is initially "EPSONKEY". Then generate a SHA256 hash of the whole string and encode the binary result in base64.

Example in pseudo code:

```
hash = encodeBase64(sha256(challenge + "EPSONKEY"))
```

Then call the login function for Administrator:

Example:

```
"function": "AuthenticateUserForAdmin",
"input": {
  "userId": "Administrator",
  "pin": "12345",
  "hash": "{base64 encoded hash}"
},
```

7.3.3 Set your own Shared Secret

Now you are logged in as Administrator in the TSE. You can now change the shared secret for the login process to something of your choice. You can use this mechanism to tie the TSE-Host to your company, a specific software or the customer. Please note that the shared secret is set per TSE on the host.

Example:

```
"function": "RegisterSecretKey",
"input": {
  "secretKey": "{your shared secret}"
},
```

7.3.4 Register client IDs

Now you can register one or more client IDs for tills using the specific TSE:

```
"function": "RegisterClient",
"input": {
  "clientId": "POS5_SHOP23"
},
```

7.3.5 Set Timeouts

There are timeouts specified that will trigger automatic actions. You can now optionally adjust those timeouts to your needs. Please note that the timeouts are being set per TSE on the TSE-Host. The following timeouts are available:

Type of Timeout	Purpose	Default Value
Admin	If there is no activity on an Admin login, Admin will be logged out.	900 seconds (15 minutes)
TimeAdmin	If there is no activity on a TimeAdmin login from a specific client id, TimeAdmin from that client id will be logged.	28800 seconds (8 hours)
Export	If the TSE is in export mode and there is none of the export function called, the export will be cancelled	100 seconds (1 minute 40 seconds)

Example:

```
"function": "SetTimeOutInterval",
"input": {
  "timeoutIntervalForAdmin": "900",
  "timeoutIntervalForTimeAdmin": "28800",
  "timeoutIntervalForExport": "100",
},
```

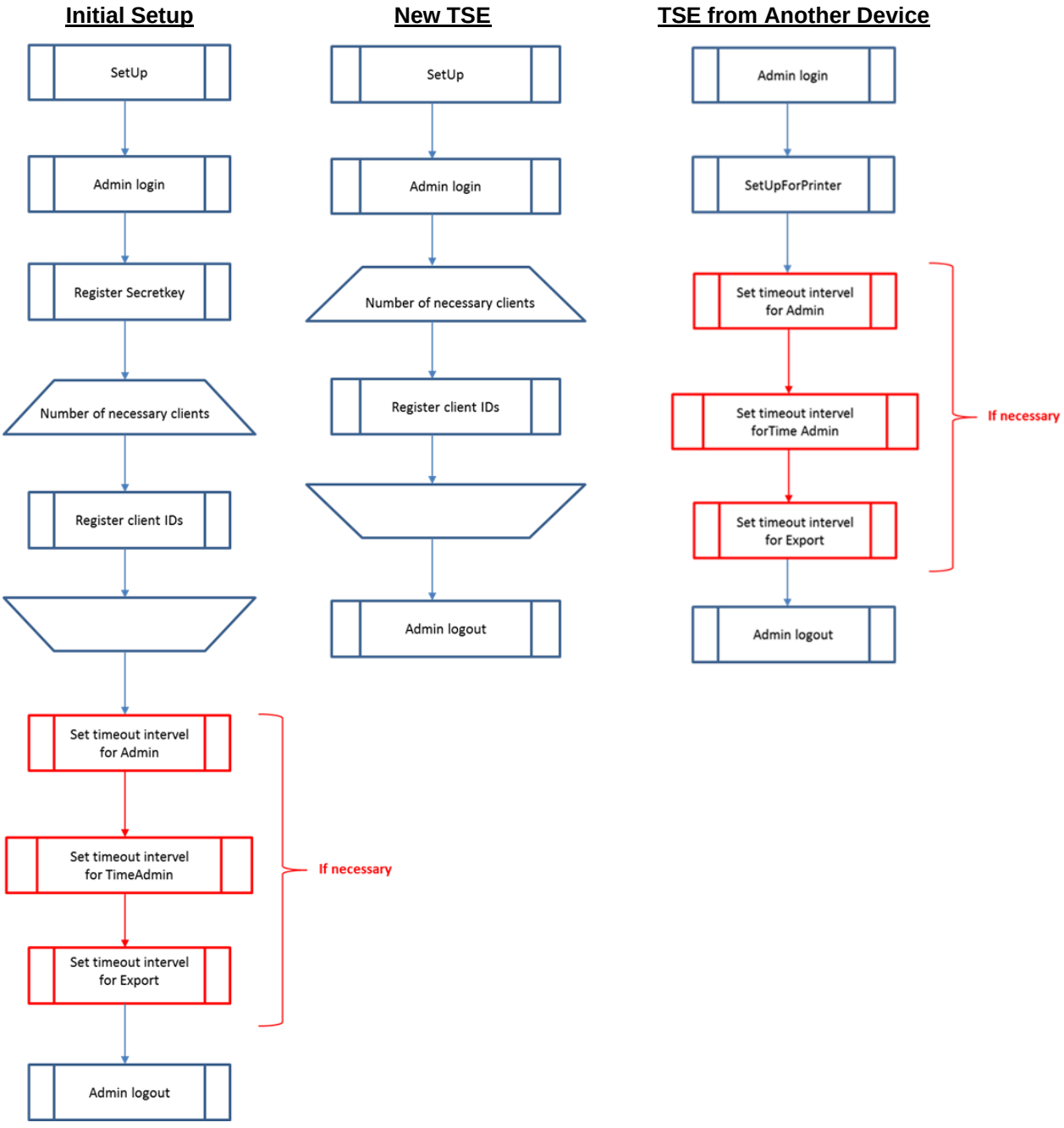
7.3.6 Logout the Administrator

At the end of the initialization process, you can logout from the Admin role.

```
"function": "LogOutForAdmin",
"input": {},
```

7.3.8 Diagrams

The following diagrams show the flow for the initial setup. Additionally, there are also diagrams on how to initialize if the TSE has been swapped in an TSE-Host and if the TSE-Host has been swapped but the TSE stays the same.



7.4 Daily operation

The flow for the daily operation is recommended should be changed when there is no other choice. Especially the TimeAdmin login should be permanent through the whole session, as the UpdateTime process only works as long at least one client is logged in.

7.4.1 Login as TimeAdmin to the TSE

In addition to the PINs, the login process is secured by a simple challenge-response mechanism. First you need to request a challenge string from the TSE-Host:

Example:

```
"function": "GetChallenge",
"input": {
  "userId": "POS5_SHOP23"
},
```

The response object has a field “challenge”, which contains a 16 characters long random string. Now concatenate the string with the shared secret, which was defined in the initialization process. Then generate a SHA256 hash of the whole string and encode the binary result in base64.

Example in pseudo code:

```
hash = encodeBase64(sha256(challenge + sharedSecret))
```

Then call the login function for TimeAdmin:

Example:

```
"function": "AuthenticateUserForTimeAdmin",
"input": {
  "clientId": "POS5_SHOP23",
  "pin":"54321",
  "hash":"{base64 encoded hash}"
},
```

7.4.2 Set the Time of the TSE

The TSE has an internal clock, that needs to be synchronized with the POS. After a power cycle the TSE will not accept any transaction before the time has been set. Please remember that the TSE works on UTC time.

Example:

```
"function": "UpdateTimeForFirst",
"input": {
  "userId": "POS5_SHOP23",
  "newDateTime": "2020-01-01T05:23:17Z",
  "useTimeSync": "false"
},
```

If more than one client connects to the TSE, only the time of the first client will be used. Subsequent calls from other clients will be ignored. If you want to explicitly refresh the time, you can use the `UpdateTime` command. Please find mere detail in the chapter "The UpdateTime Process".

The TSE-Host will use its own internal clock to regularly update the time of the TSE internally to prevent the clocks from deviating.

7.4.3 Send a Transaction

To log the required information (please refer to official documents by Bundesministerium der Finanzen / German Ministry of Finance and Bundeszentralamt für Steuern / Federal Central Tax Office) the transaction commands come in place.

A transaction consists of a start, one or more optional updates and a finish.

This an example of a transaction start:

```
"function": "StartTransaction",
"input": {
  "clientID": "POS5_SHOP23",
  "processData": "{the data to be logged in base64}",
  "processType": "{process type}",
  "additionalData": "{optional data in base64}"
}
```

The output of the command will look like this:

```
"output": {
  "result": "EXECUTION_OK",
  "logTime": "{timestamp}",
  "transactionNumber": "1",
  "serialNumber": "{certificate serial}"
  "signatureCounter": "5",
  "signature": "{signature in base64}"
},
```

The timestamp in the field `logTime` marks the start time of a sales process, which needs to be printed on the receipt.

The transaction number must be used as a parameter in subsequent `UpdateTransaction` and `FinishTransaction` calls to identify the transaction to update or finish. The TSE can hold 512 unfinished transaction in parallel, if that number is reached, it will decline any further StartTransaction call. You need to make sure to finish any transaction at some stage. You can retrieve a list of unfinished transactions from the TSE using the command `GetStartedTransactionList`.

After starting a transaction, you can update it like this:

```
"function": "UpdateTransaction",
"input": {
  "clientID": "POS5_SHOP23",
  "transactionNumber": "1",
  "processData": "{the data to be logged in base64}",
  "processType": "{process type}",
}
```

At the end of a transaction, you must finish it:

```
"function": "FinishTransaction",
"input": {
  "clientID": "POS5_SHOP23",
  "transactionNumber": "1",
  "processData": "{the data to be logged in base64}",
  "processType": "{process type}",
}
```

The output of the command will look like this:

```
"output": {
  "result": "EXECUTION_OK",
  "logTime": "{timestamp}",
  "signatureCounter": "5",
  "signature": "{signature in base64}"
},
```

The field `logTime` marks the end time of the sales process, which also needs to be printed on the receipt. The field `signature` contains the signature that should be printed on a receipt (currently not mandatory, but a planned change of regulations for receipts).

7.4.4 Self-Test

The TSE needs to perform a self-test on a regular basis in order to ensure the proper working of the signature functionality. On power up of the TSE-Host, the self-test is performed internally. After 25 hours the status “hasPassedSelfTest” in the GetStorageInfo will be false and most functions will fail with “TSE_ERROR_WRONG_STATUS_SELFTEST_NEEDED”.

In this situation please issue the following command:

```
"function": "RunTSESelfTest",
"input": {},
```

After performing the self-test, you need to update the time again.

7.4.5 Logout

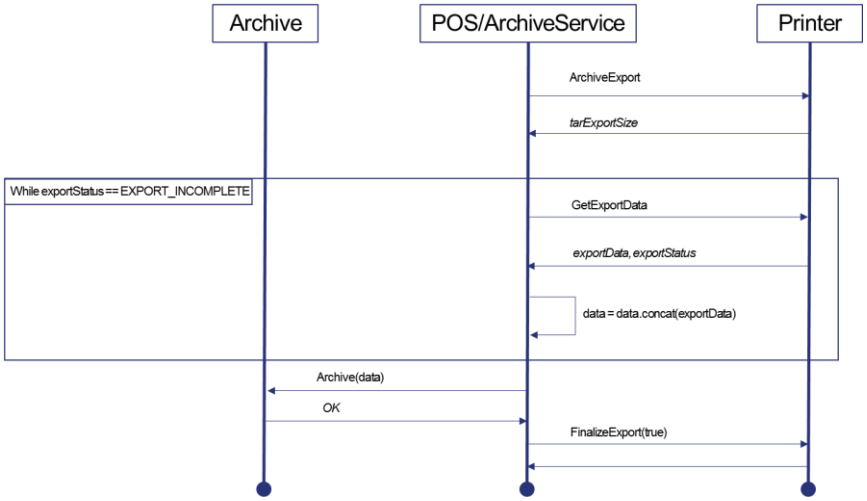
At the shutdown of the POS, the POS should logout from the TSE using

```
"function": "LogOutForTimeAdmin",
"input": {
  "clientId": "POS5_SHOP23"
},
```

7.5 Export

The data on the TSE needs to be exported to a secure archive and must be stored for 10 years. After the has been successfully exported and archived, the data can be deleted from the TSE.

The EPSON TSE API provides two types of export functions: “`ArchiveExport`” and “`ExportFilteredBy…`”.



7.5.1 ArchiveExport

This function provides all stored log messages from TSE and provides the means to delete after successful archiving. As soon you call “`ArchiveExport`” the TSE goes into export mode and will only accept the functions “`GetExportData`”, “`FinalizeExport`” or “`CancelExport`”.

Example:

```
"function": "ArchiveExport",
"input": {},
```

The response will contain the size of data that will be sent.

Example:

```
"output": {
  "result": "EXECUTION_OK",
  "tarExportSize": "186880"
},
```

7.5.2 ExportFilteredBy…

The “ExportFilteredBy…”-functions are to be used to retrieve dedicated log messages directly from the TSE (e.g. if an auditor requires data not archived yet). These functions are quite slow and do not enable deletion of exported log messages, so they should not be used for archiving. The flow for the filtered export functions is the same as with “ArchiveExport”. Please consult the JSON reference in the SDK manuals for further information (see Appendix: Related Documents).

7.5.3 GetExportData

To retrieve the data, you need to call “GetExportData”, which polls the data from the TSE.

Example:

```
"function": "GetExportData",
"input": {},
```

Due to technical limitations, only a maximum of 64 Kbytes can be transmitted in one piece. The result contains a field “exportStatus”, which has the value “EXPORT_INCOMPLETE” if more data is available and “EXPORT_COMPLETE” if all data has been retrieved.

Example:

```
"output": {
  "result": "EXECUTION_OK",
  "exportData": "{base64 encoded binary data}",
  "exportStatus": "EXPORT_COMPLETE|EXPORT_INCOMPLETE"
},
```

7.5.4 FinalizeExport

When the export has been completed, the TSE must be brought out of export mode by calling “FinalizeExport”. If the parameter “deleteData” is set to true, all log messages will be deleted from the TSE.

7.5.5 CancelExport

If there is a reason to cancel the export when it is still incomplete you can use “CancelExport” to bring the TSE out of export mode. The data on the TSE is left unchanged.

```
"function": "CancelExport",
"input": {},
```

8 Appendix: Best Practice

8.1 Order of Control

For the daily operation, the TSE should be used in the following manner to achieve the best performance. Please note that if you have multiple clients accessing the TSE, you need to connect and disconnect before and after each operation to keep the TSE available to the other clients. Performance wise it is of course better to stay connected to the TSE, which should be the practice in single client scenarios.

8.1.1 Start of the Application

- TSE connect
- GetChallenge
- AuthenticateUserForTimeAdmin
- UpdateTimeForFirst
- TSE disconnect (Multi client)

8.1.2 Signing (for short term transactions)

A short-term transaction is for example a purchase at the till in a store.

- TSE connect (Multi client)
- StartTransaction
- FinishTransaction
- TSE disconnect (Multi client)

8.1.3 Signing (for long term transactions)

A long-term transaction is for example table orders and final payment in a restaurant.

8.1.3.1 Variant 1 (process on one TSE)

On first order

- TSE connect (Multi client)
- StartTransaction
- TSE disconnect (Multi client)

On subsequent order

- TSE connect (Multi client)
- UpdateTransaction
- TSE disconnect (Multi client)

On payment

- TSE connect (Multi client)
- FinishTransaction
- TSE disconnect (Multi client)

8.1.3.2 Variant 2 (process over various TSE, needs id in process data to link transactions)

On first order

- TSE connect (Multi client)
- StartTransaction
- FinishTransaction
- TSE disconnect (Multi client)

On subsequent order

- TSE connect (Multi client)
- StartTransaction
- FinishTransaction
- TSE disconnect (Multi client)

On payment

- TSE connect (Multi client)
- StartTransaction
- FinishTransaction
- TSE disconnect (Multi client)

8.1.4 Shutdown of the Application

- TSE connect (Multi client)
- LogOutForTimeAdmin
- TSE disconnect

8.2 Status Check

A status check can take some time depending on the connection type of the TSE. It is recommended to just try an operation and then react according the error message.

The following example explains this. The situation is that the POS was not used for longer time (e.g. left powered on overnight). So, the 8-hour timeout for the TimeAdmin login expired and the last self-test was done more than 25 hours ago:

StartTransaction returns OTHER_ERROR_TIMEADMIN_NOT_AUTHORIZED

⇨ TSE needs a re-login.

AuthenticateUserForTimeAdmin returns TSE1_ERROR_SELFTEST_NEEDED

⇨ TSE needs a self-test before login

RunSelfTest return EXECUTION_OK

AuthenticateUserForTimeAdmin returns EXECUTION_OK

StartTransaction returns EXECUTION_OK

8.3 Performance

As signatures can take some time, 200 – 500ms depending on the host and the connection speed, it is advisable to have the transaction functions called in a background thread in order to keep the POS responsive.

Here an example for the short-term transaction:

1. Till operator scans first item
2. Trigger StartTransaction on thread 2
3. Take next scanned items
4. Process payment
5. Wait for StartTransaction (done by now, but good practice)
6. Trigger FinishTransaction on thread 2
7. Wait for FinishTransaction (needed, because we need the signature for the receipt)
8. Print the receipt (or generate eReceipt)

If you design your processes according to the example, there should be only a minimal performance loss adding the transaction signing to your sales processes.

9 Appendix: TSE Guidance

9.1 Initial Operation

Before an Epson TSE can be used with a POS application, a few steps must be executed.

These steps shall be executed by the POS users, so you, as POS manufacturer must guide your customers to do so.

First, the initial PINs and PUK must get changed. This is being done using the SetUp function as described in the chapter Initialization. The TSE-Host derives the initial PINs and PUK from the TSE and changes them to the values provided in the call to the SetUp function.

Finally, it is crucial that you ensure, that the user of the Epson TSE registers the TSE at the Finanzamt. This is specified in the Gesetz zum Schutz vor Manipulationen an digitalen Grundaufzeichnungen, § 146a (4). To do so, the Finanzamt will provide a form (amtlich vorgeschriebener Vordruck) that must be filled by the POS user. Epson suggests you provide a software module in your POS application to collect all required information from the user and from the Epson TSE and print them. In addition, your guidance manual must guide the user to register the Epson TSE in time. This way, the user is easily able to fill out the form and register the Epson TSE as required. The list of data which must be provided is listed in fiscal law.

In addition to the initialization of the Epson TSE itself and its registration, it is crucial that the device reaches its POS user in an unmodified form. Finally, the user must provide a secure operational environment for the Epson TSE to protect its integrity. These two aspects must be realized by your delivery process of the Epson TSE to the user and your guidance manuals, which advise the user how to set up the operational environment to protect the Epson TSE. More details on these topics can be found in the Chapter Operational Environment and Delivery to ERS-Users of this document.

9.2 The UpdateTime Process

The internal clock of the TSE needs be corrected after a specific time span, to correct any irregularities. Once the TSE-Host has been provided with a valid time using "`UpdateTime`" or "`UpdateTimeForFirst`", it will take care of regularly updating the time of the TSE until the TSE-Host restarts.

The reason there are two functions to update the time is to provide you the choice on how to manage some kind of "master-time" if more than one client is using one TSE.

"`UpdateTimeForFirst`" will update the time only, if it hasn't been set by another client yet. If another client has already set the time and the UpdateTime process is still active, the subsequent calls to "`UpdateTimeForFirst`" will return "`EXECUTION_OK`" but will not change the time for the TSE. Otherwise it will set the TSEs time and start the UpdateTime process.
By just using this function it is made sure that always the first client logging in to a TSE will define the TSE time.

"`UpdateTime`" will update the time in any case. If not done yet, it will also start the UpdateTime process. If you plan to manage the "master-time" between your clients by yourself, then this is the function to use.

If there is only one client using the TSE, it does not matter which of both commands you use.

9.3 PIN and PUK Handling

If you decide to change the PIN (and PUK) values prior to delivery to your customers, make sure (by a corresponding implementation in your POS application), that the POS user has to change the PIN (and PUK) values again. It is not adequate that you are aware of the PIN (and PUK) values of a TSE, which is in use by a POS user!

Finally, do not store the values of the Admin PIN or PUK within the software of your electronic cash register! These values are credentials of the owner of the Epson TSE and should be known to him/her only. Ensure that the POS user is aware of this by documenting it accordingly in the POS user's guidance manual. In addition, add remarks that ensure, that the POS user is aware of his/her surrounding while changing PINs to prevent somebody else watching the new PIN values being entered by the POS user. Storing the TimeAdmin PIN in the POS software is acceptable though.

9.4 Life Span of an Epson TSE

Two aspects determine the life span of an Epson TSE:

- the number of signatures, its cryptographic service provider computed and
- the validity of the certificate of the cryptographic service provider.

The cryptographic service provider can create twenty Million (20.000.000) signatures. When an Epson TSE has created this number of signatures it is strongly suggested to replace the TSE.

The validity of the certificate depends on the point of time, when the certificate was created. This happens as part of the production process and the Epson TSE does not support to switch certificates after production.

Both information, the number of created signatures and the validity of the certificate can be read using the function GetStorageInfo.

For common usage, the Epson TSE has more than enough memory to store all transactions which will be made in its lifetime. In special cases, especially if each transaction contains a lot transaction data, it might be the case, that the Swissbit TSE runs out of memory. Again, the function GetStorageInfo contains information about the TSE's capacity and the current size of Logs. If the current size comes too close to the capacity, either the TSE should be replaced, or the log files should be exported and stored safely. Afterwards, the Epson TSE Logs can be deleted on the TSE by the Administrator and the TSE be used further. Please find details on that in the chapter Export.

9.5 Information on long term operation

This information is important when the TSE-Host is running for a long time or is running 24/7.

9.5.1 Self-Test

25 hours after the last self-test the TSE loses its self-test-passed-status. Then a self-test must be performed. The self-test also resets the TSE clock, so a new UpdateTime(ForFirst) is also necessary.

9.5.2 Login Timeouts

For Admin and TimeAdmin logins login timeouts exist. By default, Admin logins will be void after 15 minutes, TimeAdmin logins after 8 hours of inactivity on that login.

Those timeouts, as well as the timeout for an export to be cancelled can be checked using `GetTimeOutInterval` and changed with `SetTimeOutInterval`.

10 Appendix: Troubleshooting

10.1 All TSE-Hosts

10.1.1 TSE1_ERROR_CLIENT_NOT_REGISTERED occurs

10.1.1.1 General

A ClientID was used for a transaction command which is not registered on the TSE. Please use `RegisterClient` to register the ClientID or use another, registered ClientID.

10.1.1.2 Transaction command yields the error although logged in

By the BSI specification, the TSE knows just the user roles Admin and TimeAdmin and each role has its PIN. The ClientID is only passed to `AuthenticateUserForTimeAdmin` to make it possible for the TSE-Host to keep track of the clients currently using the TSE.

`AuthenticateUserForTimeAdmin` does not check if the ClientID is registered, but a signature can only be created with a registered ClientID.

10.1.1.3 Error occurs on RunTSESelfTest

Even though `RunTSESelfTest` does not take a ClientID as parameter, it uses internally the ClientID that the TSE-Host registers internally on `SetUp`.

This error can happen in the two following situations:

The TSE is not initialized yet

The TSE cannot have a registered ClientID yet. After running `SetUp`, the self-test will run without error.

The TSE was moved to another TSE host

The ClientID for the new host must be registered. Please do this using `SetUpForPrinter`.

10.1.2 SetUp fails

If any of those situations below happen, make sure not to retry with different PUK or PINs. This could render the TSE unusable.

10.1.2.1 SetUp fails with TSE1_ERROR_INVALID_PARAMETER

This error can only happen on a PC with a Windows driver older than version 1.0.5.0. The host name of the PC contains an underscore, which cannot be used in a client ID. Please update the Windows driver to the newest version. Otherwise change the host name of the PC. **Please make sure to use the same PUK and PINs in the next call of `SetUp`.**

10.1.2.2 SetUp fails with TSE1_ERROR_SIG_ERROR or TSE1_ERROR_IO_ERROR

To recover, please make sure the TSE power cycles (by restarting the printer, removing or reinserting the TSE in the PC or TSE Server). **Please make sure to use the same PUK and PINs in the next call of `SetUp`.**

If this error happens, please check the version of the firmware of the printer. Make sure to update the printer to the latest version. It is recommended to have the printer on the latest firmware version anyway

10.1.3 OTHER_ERROR_CURRENTLY_EXPORTING after reconnection

This error occurs when a different command as `GetExportData` or `CancelExport` are being sent while the host is exporting. If the connection to the TSE was lost for some reason, after reconnection the first thing should be a `CancelExport` and then a retry to export.

If that also fails, the other solution is to wait until the export timed out (see command `GetTimeOutInterval`).

10.1.4 OTHER_ERROR_HOST_AUTHENTICATION_FAILED

Please make sure that you use the correct secret key for the current host. Please note that the secret key is linked to the TSE-Host, not to the TSE.

For each `Authenticate...` command request a new challenge. A challenge can only be used once, even if the `Authenticate...` command failed for some reason.

10.1.5 OTHER_ERROR_UNAUTHENTICATED_ADMIN_USER OTHER_ERROR_UNAUTHENTICATED_TIME_ADMIN_USER

If the error comes surprisingly, then the user was automatically logged out. There are inactivity timeouts for the Admin and TimeAdmin login out (see command `GetTimeOutInterval`).

10.1.6 OTHER_ERROR_UNKNOWN_STORAGE

Please check if the TSE is connected correctly. If it is a USB TSE, please make sure it is plugged into the USB port correctly, in case of a microSD TSE make sure it is inserted correctly into the microSD slot of the printer or the adapter OT-UH30. If the adapter is used, please make sure it plugged into the USB port of the printer correctly and the screw to fasten the adapter is fastened.

If this error still occurs, please return the TSE to Epson via your distributor.

10.1.7 OTHER_ERROR_NO_STORAGE_FOUND

Please check if the TSE is connected correctly. If it is a USB TSE, please make sure it is plugged into the USB port correctly, in case of a microSD TSE make sure it is inserted correctly into the microSD slot of the printer or the adapter OT-UH30. If the adapter is used, please make sure it plugged into the USB port of the printer correctly and the screw to fasten the adapter is fastened.

If this error still occurs, please return the TSE to Epson via your distributor.

10.1.8 TSE1_ERROR_SELF_TEST_FAILED_CSP

If this error occurs, please return the TSE to Epson via your Epson TSE supplier.

10.2 Driver Specific

10.2.1 TSE1_ERROR_NOT_AUTHORIZED

If this error occurs, please update the driver to the latest version.

10.2.2 Windows claims TSE is not formatted

If this error occurs when the TSE is definitely connected correctly, please return the TSE to Epson via your Epson TSE supplier.

10.2.3 TSE Monitor Icon is Green, but no TSE is connected

Please check the PC if there is any further storage device with the name EPSON is connected. This can be a USB stick with the volume name EPSON or maybe an Epson driver CD.

Please remove those devices or disks from the PC and re-insert the TSE.

10.2.4 OTHER_ERROR_UNKNOWN_STORAGE

Please check the PC if there is any further storage device with the name EPSON is connected. This can be a USB stick with the volume name EPSON or maybe an Epson driver CD.

Please remove those devices or disks from the PC and re-insert the TSE.

In the case only a TSE is connected to the PC, please contact tour Epson TSE supplier for support.

10.3 Requesting Support

When asking for technical support or returning a TSE, always refer to your Epson TSE supplier first and please provide as much information as possible - the more information the faster the support!

10.3.1 Logs

10.3.1.1 Application Logs

Please provide logs of your application, in which it is visible which commands had been sent to the TSE and what was the result.

10.3.1.2 EPS TSE Server Logs (if applicable)

If you are using the EPS TSE Server please download the server logs via the web UI or the SEH Product Manager.

10.3.1.3 Windows Driver Logs

Please provide the contents of the folder [ProgramData]/EPSON/TSE as a zip file. If the issue is reproducible, please increase the driver's log level by opening the file [ProgramData]/EPSON/TSE/TSEDrvConf in a text editor and change the value for the LogLevel to 3.

10.3.2 Questionnaire

When contacting support, please provide the answers to the following relevant questions to assure a quick response.

10.3.2.1 TSE Information:

Type of TSE:	USB, microSD
Type of TSE-Host	Printer, EPS TSE Server, PC
Output of GetStorageInfo:	e.g. exported via TSE Monitor in the driver
TSE_INFO.DAT:	File should be copiable from the TSE. If not, please note this fact.
Lot number:	Printed on the TSE

10.3.2.2 Design Phase & Special Condition

Development:	Yes or No	Did the problem occur during development?
Stress Test:	Yes or No	Did the problem occur during a stress test?
Field Failure:	Yes or No	Did the problem occur in productive use?

10.3.2.3 Environmental conditions:

Exceptional temperature: Yes or No

If yes: Hot or Cold if available: average, highest/lowest temperature

High humidity: Yes or No

10.3.2.4 Development Information:

Programming language: (C, C++, Objective C, Swift, C#, VB.NET, Java etc.)

Framework: (e.g. Xamarin)

10.3.2.5 System Information:

Type of TSE-host:	The device the TSE is directly connected to: Printer, EPS TSE Server or PC

Printer

Model:	TM-T88VI-iHub, TM-m30F, TM-m30II
Interface:	USB, Ethernet, Wi-Fi, Bluetooth, Serial
POS-Platform:	PC, Mobile Device, Web
POS-OS:	Windows, Linux, Android, iOS, Browser (incl. details like version, distribution etc.)
SDK:	ePOS SDK, ePOS Device XML, Windows SDK (incl. Version), ESC/POS

EPS TSE Server

Model:	3 Port or 8 Port
POS-Platform:	PC, Mobile Device, Web
POS-OS:	Windows, Linux, Android, iOS (incl. details like version, distribution etc.)
SDK:	ePOS SDK, ePOS Device XML, Windows SDK (incl. Version)

PC:

OS:	Windows, Linux (incl. version, distribution)
Driver version:	All version numbers listed in Epson TSE Monitor
SDK:	ePOS Device XML, Windows SDK (incl. Version)

10.3.2.6 Details on the Issue:

Description:	What happens? What is expected?
Steps to reproduce:	All steps since power on of the TSE and POS
Actual error:	Error code, Screen shot with error message... Whatever is available

11 Appendix: Related Documents

11.1 Epson Online Resources

11.1.1 Tools, Drivers and Sample Code for the Epson TSE

https://download.epson-biz.com/modules/pos/index.php?page=genre&pcat=51

11.1.2 The SDKs

https://download.epson-biz.com/modules/pos/

On the software tab you can find the links to the SDKs.

11.1.3 Online Technical References for all SDKs

https://reference.epson-biz.com/pos/reference/

An offline manual can be found in the SDK packages from the download section.

11.2 Official Government Legislation Documents

11.2.1 General Technical Regulations for the TSE by BSI

(Bundesamt für Sicherheit in der Informationstechnik, Federal Office for Security in Information Technology)

11.2.1.1 TR-03153

A document describing the technical requirements for a TSE (German language)

https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03153/index_htm.html

11.2.1.2 TR-03151

A document describing the requirements for the SE-API of a TSE (English language)

https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03151/index_htm.html

11.2.2 Fiscal Regulations by BMF

(Bundesministerium für Finanzen / Federal Ministry of Finance)

For all questions regarding legal topics or fiscal compliance please refer to BMF directly via IVA4@bmf.bund.de.

11.2.2.1 Anwendungserlass zu §146a AO

(Circular on Application of the Fiscal Code)

https://www.bundesfinanzministerium.de/Content/DE/Downloads/BMF_Schreiben/Weitere_Steuerthemen/Abgabenordnung/AO-Anwendungserlass/2019-06-17-einfuehrung-paragraf-146a-AO-anwendungserlass-zu-paragraf-146a-AO.html

11.2.2.2 DSFinV-K 2.1

Description of data to be recorded on a POS (German only).

https://www.bzst.de/DE/Unternehmen/Aussenpruefungen/DigitaleSchnittstelleFinV/digitaleschnittstellefinv_node.html