



# Kofax Equitrac Campus Card API Reference Guide

Version: 6.5.0

Date: 2024-04-18

© 2011– 2024 Tungsten Automation. All rights reserved.

Tungsten and Tungsten Automation are trademarks of Tungsten Automation Corporation, registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Tungsten Automation.

# Table of Contents

Campus Card system Overview.....	5
What is a Campus Card API?.....	5
Concepts, terms, and abbreviations.....	6
Equitrac and Campus Card interaction.....	8
General usage.....	8
System Manager interaction.....	8
CAS interaction.....	9
Creating your own interaction.....	10
API package contents.....	10
Getting started.....	10
Programming language.....	11
Implementing the DLL.....	11
Sample projects.....	12
Deployment.....	13
API reference.....	15
Enumerations.....	15
EQFASAPIRetCode.....	15
EQFASAPIEntryMethod.....	16
Structures.....	16
EQFASAPIINFO.....	16
EQFASAPITENDERKEYMAP.....	17
EQFASACCOUNTINQUIRYPARAMS.....	17
EQFASTRANSACTIONPARAMS.....	18
Methods.....	19
eqFAS_InitializeDLL.....	19
eqFAS_TerminateDLL.....	20
eqFAS_GetInfo.....	21
eqFAS_FreeInfo.....	21
eqFAS_SetConfiguration.....	21
eqFAS_GetConfiguration.....	22
eqFAS_FreeConfiguration.....	23
eqFAS_SetTenderKeyMap.....	23
eqFAS_GetTenderKeyMap.....	24
eqFAS_FreeTenderKeyMap.....	24

- eqFAS\_ShowConfigurationUI..... 25
- eqFAS\_Connect..... 25
- eqFAS\_Disconnect..... 26
- eqFAS\_AccountInquiry..... 26
- eqFAS\_DebitAccount..... 27
- eqFAS\_RefundAccount..... 28
- eqFAS\_DepositAccount..... 28
- Test application..... 30
  - Prerequisites..... 30
  - Start the test application..... 30
  - Use the test application..... 30

# Campus Card system Overview

In an Equitrac solution, Campus Card systems refer to those capable of storing user accounts and money balance for both themselves and Equitrac. Campus Cards systems are also referred to as Foreign Accounting Systems (FAS). Equitrac must be able to communicate with Campus Card systems to enable integration with card payment systems.

A typical scenario would include a company with an existing system that stores accounts and balances, which the company plans to use with Equitrac. As a result, the Campus Card system assumes responsibility for storing money instead of the Equitrac system. If both the Campus Card system and Equitrac store accounts and balances, this is also a working solution. Additionally, a Campus Card account may have several tender keys. See [Tender keys](#).

If a user walks up to a controller or device and authenticates herself as a Campus Card account holder, the cost of operation (scan, copy, fax or print) is debited from the Campus Card account.

Equitrac accounts can also be assigned to Campus Card accounts. In this case, the Campus Card account has the same functionalities (rules, price list, etc.) as the Equitrac account.

During user authentication, or when there is a decrease in the Campus Card account balance, the Equitrac server communicates with the Campus Card system according to Equitrac's System Manager configuration settings.

Equitrac and Campus Card systems are different applications using different technologies. This integration should be made for the Campus Card system being used.

## What is a Campus Card API?

The Campus Card Application Programming Interface (API) is an Equitrac API specifically developed for integrating different Campus Card systems. This API defines entry points called by Equitrac during authentication and payment processes. Integrators should create a module implementing the entry points of the Campus Card API, use the Campus Card system to perform the requested operations and send the results back to Equitrac.

The Campus Card API is provided to integrators in a package, which includes the definition of the entry points, sample projects to ease development and a test application to test the integration.

## Concepts, terms, and abbreviations

For a full understanding of this API documentation, the following concepts, terms and abbreviations must be clarified:

### **MFD**

Multifunction Device; used for common office purposes, such as printing, scanning, faxing and copying.

### **SFD**

Single Function Device; used only for one purpose, mostly printing.

### **CAS**

Core Accounting Server; a central part of the Equitrac server suite responsible for storing accounts, performing authentication, calculating transaction charges and assigning those charges to appropriate users or group accounts. Each Equitrac component communicates with CAS.

### **DCE interface**

Device Control Engine; a controller or embedded device client controlling SFDs or MFDs and providing the user interface to end users. DCE interfaces communicate with the Device Control Engine (DCE) component of Equitrac to perform authentication, unlock the assigned device and start an operation on the device.

### **Equitrac account**

An account (user, department or billing code) stored and handled by Equitrac. Based on an Equitrac account, several Equitrac features (limit quotas, rules) are available.

### **Campus Card account**

An account stored by the Campus Card system. A Campus Card account may have several tender keys; in this case CAS handles them as separate Campus Card accounts. A Campus Card account inside CAS may have an account name, which can be used to identify the print jobs in the print queue.

### **Network ID of a Campus Card account**

An optional property of a Campus Card account, which stores an account name. This account name can be a Windows account name or an Equitrac account user ID. It is used to assign a Windows or Equitrac account to a Campus Card account to identify the available print jobs from the print queue. Not all Campus Card systems support this feature. The Campus Card system administrator is responsible for setting it.

### **Campus Card mapping table**

A database table storing the primary identification data (mainly card number) and an account name to assign a Windows or Equitrac account to a Campus Card account. This table is dynamically updated when CAS acquires Campus Card accounts. Additionally, data can be imported into this table with the CampusCardImport.exe utility, located in the Tools subfolder of the Equitrac installation folder (C:\Program Files\Nuance\Equitrac\Tools).

### **Campus Card offline cache**

Communication between Equitrac and the Campus Card system might be disrupted (network issue or maintenance). As a workaround, Equitrac is able to store the authenticated Campus Card accounts and their balances in a cache, utilized if Equitrac cannot communicate with the Campus

Card system. This offline cache has a maximum balance (for all stored accounts), which can be used by the offline transactions. The Campus Card offline cache can be configured in System Manager.

**Tender keys**

Location for storing the user-specific money amounts, similar to a purse or wallet. Not all Campus Card systems have this functionality, therefore some Campus Card accounts have only one tender key. Every Campus Card account must have at least one tender key.

**Campus Card integrator DLL**

A DLL implementing the Campus Card API and providing a bridge between Equitrac and a Campus Card system. This document details how to implement such a DLL.

**Foreign Accounting System (FAS)**

Another name for Campus Card system in Equitrac.

# Equitrac and Campus Card interaction

## General usage

Before implementing a Campus Card integration, you should understand how Equitrac uses Campus Card systems.

A Campus Card system must be configured in Equitrac's System Manager. Once configured, the Equitrac Core Accounting Server (CAS) starts using the Campus Card system for device operations during user authentication and when paying for operations at SFDs/MFDs. See [System Manager interaction](#) and [CAS interaction](#).

**i** Only one Campus Card system can be configured for one CAS.

Campus Card systems can be integrated with Equitrac by developing a DLL (Campus Card Integrator DLL), which implements the Campus Card API of Equitrac and communicates with the Campus Card system. See [Creating your own interaction](#).

When Equitrac wants to communicate with a Campus Card system, System Manager or CAS calls into the Campus Card Integrator DLL using the entry points of the Campus Card API. See [API reference](#).

## System Manager interaction

The Campus Card system can be configured in System Manager. During configuration, System Manager communicates with the Campus Card system by calling into the Campus Card Integrator DLL.

When the Campus Card Integrator DLL is copied to the proper location (see [Deployment](#)), System Manager automatically recognizes and displays it in the Foreign Accounting System drop-down list of the Campus Card interface dialog (see the *Kofax Windows System Manager Guide* for more details about this dialog). Here the Campus Card system can be configured (for example, setting the connection details) for Equitrac.

System Manager uses the Campus Card Integrator DLL in the following manner:

- When the Campus Card interface dialog is opened:
  1. The DLL is loaded into memory.
  2. The DLL is initialized.



3. Information about the Campus Card integration is acquired.
  4. The DLL is terminated.
  5. The DLL is unloaded from memory.
- The configuration of the Campus Card integration is owned and handled by the Campus Card Integrator DLL. To provide a user-friendly way of the configuration, the Campus Card Integrator DLL should contain a configuration UI (a modal dialog), where the configuration can be set. Equitrac is capable of storing the configuration settings in its database. The serialized configuration string is transferred between Equitrac and the Campus Card Integrator DLL.
  - When the Configure button is pressed on the Campus Card interface dialog:
    1. The DLL is loaded into memory.
    2. The DLL is initialized.
    3. The serialized configuration string stored by Equitrac is sent to the DLL.
    4. A list of allowed tender keys stored by Equitrac is sent to the DLL.
    5. Configuration UI of the DLL is displayed.
    6. A list of the allowed tender keys is acquired from the DLL.
    7. The serialized configuration string is acquired from the DLL.
    8. The DLL is terminated.
    9. The DLL is unloaded from the memory.

## CAS interaction

If a user walks up to a DCE interface and tries to log in (either with a card swipe or by entering identification data), the card number or the identification data is sent to CAS for user authentication. If a Campus Card system was configured in System Manager, CAS queries the Campus Card system for authentication by sending the card number or the identification data to the Campus Card Integrator DLL. The integrator DLL must use the Campus Card system to attempt and acquire a Campus Card account and respond to CAS. CAS collects all information on authentication results, recognized users, then sends them back to the DCE interface and enables user login.

Later, if the user performs a billable operation (scan, copy, fax or print) on the SFD/MFD, the DCE interface sends the operation cost to CAS. If the user was logged-in with a Campus Card account, CAS tells the Campus Card system to pay the cost of the operation by calling into the Campus Card Integrator DLL.

When the Campus Card Integrator DLL is copied to the proper location (see [Deployment](#)), CAS automatically starts using it (if it is already properly configured in System Manager).

CAS uses the Campus Card Integrator DLL in the following manner:

- When CAS is started, or the Campus Card configuration is modified in System Manager:
  1. The DLL is loaded into memory.
  2. The DLL is initialized.
  3. The serialized configuration string stored by Equitrac is sent to the DLL.

4. A list of allowed tender keys stored by Equitrac is sent to the DLL.
  5. The DLL is asked to connect to the Campus Card system.
- If the DCE interface sends user credentials to CAS and CAS tries to authenticate with the Campus Card system:
    1. The DLL is asked to find a Campus Card account for all allowed tender keys. This is done by going through all the allowed tender keys.
  - If an operation initiated by a Campus Card account has to be paid for or if the Campus Card offline cache is enabled and attempts are made to process the cached transactions:
    1. CAS asks the DLL to debit the account balance.
  - When CAS is stopped, or the Campus Card system is deselected in System Manager:
    1. The DLL is asked to disconnect from the Campus Card system.
    2. The DLL is terminated.
    3. The DLL is unloaded from memory.

## Creating your own interaction

The goal is to develop a DLL (Campus Card Integrator DLL), serving as a bridge between Equitrac and the Campus Card system. Equitrac provides the Campus Card API package with the tools needed to help with this task.

### API package contents

- *Campus Card API Reference Guide* – This reference guide in PDF format.
- `eqFASapi.h` – A C header file defining the entry points and types of the Campus Card API. The Campus Card Integrator DLL must have these public entry points and they need to be implemented. The entry points and structures are described in [API reference](#).
- `SampleProjects` folder – Contains sample projects as a starting point to develop a Campus Card Integrator DLL. They ease the development of a new Campus Card system integration as described in [Sample projects](#).
- `TestApp` folder – Contains both the 32-bit and 64-bit editions of a test application, which can be used to test a Campus Card Integrator DLL during development. See [Test application](#).

### Getting started

Before integrating the Campus Card system into Equitrac, explore it for better understanding. Examine how to use its infrastructure (business objects, web services, command line, and so forth) to perform the following actions:

- Establish a communication channel with the Campus Card system (for example, loading any of its DLLs, calling a web service or providing administrator privileges).
- If the Campus Card system supports tender keys, collect all allowed from the system.

- Get a Campus Card account (if any) based on one or optionally two identification strings (like username and password) and a specified tender key, if tender keys are supported, with the following information:
  - Current account balance.
  - Optionally, a minimum account balance.
  - Optionally, a network ID to help find a corresponding Windows or Equitrac account.
- Perform operations on the identified Campus Card account balance (if any), based on one or optionally two identification strings (like username and password), and a specified tender key, if tender keys are supported. After the operation has been performed, the new balance is returned. The required operations are the following:
  - The account is debited with a specified amount to decrease its balance.
  - Optionally, a specified amount is deposited to the account to increase its credit balance.
  - Optionally, a specified amount is refunded to the account to restore its credit balance.

## Programming language

In Equitrac, both System Manager and CAS (wishing to communicate with the Campus Card system) are native C++ applications developed using the C++ programming language, thus the Campus Card API is defined as a standard C header file. The Campus Card Integrator DLL has to be a DLL, which has the defined entry points and can be called from a C application.

Additionally, Equitrac uses the Microsoft Visual C++ 2010 SP1 runtime (download at <http://www.microsoft.com/en-us/download/details.aspx?id=8328> or <http://www.microsoft.com/en-us/download/details.aspx?id=13523>), therefore it is already installed on the computer where Equitrac is installed.

Because of this, the recommended technology to implement a Campus Card Integrator DLL is the standard C/ C++ with Visual Studio 2010 SP1, and in this case the integration is simply as described in [Deployment](#) and no extra steps are required.

If a different programming language or C/C++ runtime is used during integration, the used runtime library must also be shipped with the integrator DLL.

Equitrac CAS has both 32-bit and 64-bit editions. Depending on which editions are used, both 32-bit and 64-bit versions of the Campus Card Integrator DLL may be required.

- If all installations of Equitrac are 32-bit, only the 32-bit edition of the Campus Card Integrator DLL is required.
- If you use a mix of 64-bit and 32-bit instances of Equitrac CAS, both editions of the Campus Card Integrator DLL are required, because the System Manager only has a 32-bit edition.

It is best to prepare the Campus Card Integrator DLL project to be able to generate both 32-bit and 64-bit DLLs.

## Implementing the DLL

Once the base project is ready (with the selected technology), implementing the Campus Card API entry points can commence.

When using the C or C++ programming language, the provided `eqFASapi.h` C header file must be included in the project. In case of a different programming language, one should know how to import or define the same API and generate the appropriate DLL. Once ready, implement each method in the header file by communicating with the Campus Card system. The API methods are described in detail in [API reference](#).

**i** Two sample projects are provided in the API package, with the header file already imported, having a minimal implementation of the API. Starting development with these projects is recommended as described in [Sample projects](#).

**i** CAS has a time guard implemented for the Campus Card integration. The time consumed in the account inquiry and balance operations methods should not exceed 10 seconds. If the processing time is too long, CAS behaves as if the Campus Card system is in offline mode. The reason for this is that the methods described are initiated from a DCE interface, which has a built-in timeout value. If CAS spends too much time with authentication or payment for an operation, the user is rejected or logged off by the DCE interface.

During development, it is recommended to use the Campus Card test application as described in [Test application](#). With this, each API method of the integrator DLL can be called from a user interface. DLL implementation can be attempted and tested easily, without affecting a running Equitrac system. Finally, if the Campus Card Integrator DLL is ready for testing or usage in a real Equitrac environment, the DLL should be deployed into Equitrac, as described in [Deployment](#).

## Sample projects

Two sample projects are shipped with the Campus Card API package, which can be used as a starting point for the implementation.

The sample projects are created with Visual Studio 2010 using C/C++ programming language and they implement each method of the Campus Card API.

The sample projects are located in the SampleProjects folder in the API package:

- `EQFASAPI.sln` is the Visual Studio 2010 solution file containing the two projects.
- `FASSampleWithUI` folder contains the `FASSampleWithUI.vcxproj` Visual Studio 2010 C++ project file and the additional source files.
  - In case of using C++ programming language with Visual Studio to implement the Campus Card Integrator DLL, starting with this project file is strongly recommended.
  - This project has a configuration UI implemented with Microsoft Foundation Class (MFC) libraries.
  - It emulates a single user with configurable card number, balance and tender keys.
  - Some important source files:
    - `FASSampleWithUI.def`: Lists the public entry points of the DLL as required by the API.
    - `FASInterfaceProxy.cpp`: Implements the API methods by passing each call to the C++ class.
    - `FASSampleWithUI.h/cpp`: The main C++ class performing the work.
    - `FASSampleConfigDlg.h/cpp`: Implements the configuration dialog.

- `EQTenderKeyGrid.h/cpp`: Implements the grid for tender keys.
- `Tracer.h/cpp`: Helper class for tracing.
- `FASSampleNoUI` folder contains the `FASSampleNoUI.vcxproj` Visual Studio 2010 C project file and the additional source files.
  - When using the C programming language (neither with the Microsoft C compiler nor with Microsoft Visual Studio) to implement your Campus Card Integrator DLL, starting with this project file is recommended.
  - This project defines only the API methods without any specific implementation and any UI.
  - Some important source files:
    - `FASSampleNoUI.def`: Lists the public entry points of the DLL as required by the API.
    - `FASSampleNoUI.cpp`: Implements the API methods.
    - `Tracer.h/cpp`: Helper class for tracing.

The sample projects are easy to use with Microsoft Visual Studio. Simply open the solution and both sample projects are ready to be built. They are prepared to generate both 32-bit and 64-bit outputs into a bin subfolder.

Once the sample projects are built, the generated DLLs are ready to be used either in the Campus Card test application ([Test application](#)) or integrated with Equitrac ([Deployment](#)).

After the sample projects are reviewed and built, you can use the preferred sample project by implementing the communication with the Campus Card system. In this case the only task is to implement the Campus Card integration, since all other parts (for example integration into Equitrac, logging) are already prepared.


## Deployment


When the Campus Card Integrator DLL is ready to be used with Equitrac, deploy it into an installed and running Equitrac system. This must be completed by copying the DLL to the appropriate subfolders of the Equitrac installation folder (C:\Program Files\Nuance\Equitrac\ by default).

Since both CAS and System Manager use the Campus Card Integrator DLL, the integrator DLL must be deployed to each computer where either CAS or System Manager is installed. See [General usage](#).

Preferred deployment steps:

1. Locate all Core Accounting Servers, where the Campus Card integration is to be used.
2. Deploy the Campus Card Integrator DLL to each of these Core Accounting Servers. Copy the Campus Card Integrator DLL to the `Common\FAS` subfolder of the Equitrac installation folder.

 Proper integrator DLL edition(32-bit or 64-bit) must be copied to the corresponding (32-bit or 64-bit) subfolder as CAS is running.

 Restarting CAS for using the newly copied DLL is not necessary. As the Campus Card interfaceconfiguration dialog of System Manager is closed with its OK button, CAS starts using the configured Campus Card integration.

3. Locate each System Manager, being used to configure Core Accounting Servers, where the Campus Card integration is to be used.
4. Deploy the Campus Card Integrator DLL to all these System Managers.

**i** Restarting System Manager for detecting the newly copied DLL is not necessary. Every time the Campus card interface configuration dialog is opened, the available Campus Card Integrator DLLs are automatically recognized.

5. Configure the Campus Card integration in the Campus card interface dialog of System Managers. More information about this dialog can be found in the *Kofax Windows System Manager Guide*.

If the Campus Card integration becomes deprecated or before uninstalling Equitrac, it is recommended to remove the copied integrator DLLs from the Equitrac installation folder.

# API reference

## Enumerations

### EQFASAPIRetCode

Lists the available return values of the methods.

This enumeration is returned by the following methods:

- eqFAS\_InitializeDLL
- eqFAS\_TerminateDLL
- eqFAS\_FreeInfo
- eqFAS\_SetConfiguration
- eqFAS\_FreeConfiguration
- eqFAS\_SetTenderKeyMap
- eqFAS\_FreeTenderKeyMap
- eqFAS\_ShowConfigurationUI
- eqFAS\_Connect
- eqFAS\_Disconnect
- eqFAS\_AccountInquiry
- eqFAS\_DebitAccount
- eqFAS\_RefundAccount
- eqFAS\_DepositAccount

#### Available values

Value	Description
eFAS_OK	The method performed its task successfully, without any errors.
eFAS_Fail	There was an error. This value means a generic failure, but use a more specific value from below if possible.
eFAS_InvalidParameter	The method was called with an invalid parameter.
eFAS_NotInitialized	The Campus Card integration has not been initialized yet.
eFAS_AlreadyInitialized	The Campus Card integration is already initialized.

Value	Description
eFAS_CommunicationsError	There was a communication error with the Campus Card system.
eFAS_ValidationFailure	Campus Card account could not be found.
eFAS_InsufficientFunds	The Campus Card account does not have enough money to pay the requested amount.
eFAS_PartialFundsDeducted	The Campus Card account does not have enough money to pay the requested amount, but all the available money have been used.
eFAS_TransactionDenied	The Campus Card system has rejected the transaction.
eFAS_SecondaryPINRequired	Although a Campus Card account could be identified based on the received primary identifier, the Campus Card system also requires a secondary identifier for security reasons.

## EQFASAPIEntryMethod

Lists the available authentication mode used on the DCE interfaces. For instance, a Campus Card system may work only with card swipes.

This enumeration is returned by the following methods:

- EQFASTRANSACTIONPARAMS
- EQFASACCOUNTINQUIRYPARAMS

### Available values

Value	Description
eSwipe	Authentication performed by swiping a card, primary identification data is a card number.
eKeypad	User credentials were entered on the DCE interface.

## Structures

### EQFASAPIINFO

Contains information about the Campus Card integration. This information is displayed on the Campus card interface dialog of System Manager.

This structure is used by the following methods:

- eqFAS\_GetInfo
- eqFAS\_FreeInfo

### Members of this structure

Type	Name	Description
unsigned_long	cbSize	Size of structure (used for versioning).



Type	Name	Description
const_wchar_t*	pszShortName	Short name of the module.
const_wchar_t*	pszDescription	Longer description about the module.

## EQFASAPITENDERKEYMAP

Represents a tender key list to return the allowed tender keys or receive the allowed tender keys from System Manager or CAS. .

This structure is used by the following methods:

- eqFAS\_SetTenderKeyMap
- eqFAS\_GetTenderKeyMap
- eqFAS\_FreeTenderKeyMap

### Members of this structure

Type	Name	Description
unsigned_long	ulNumMapEntries	Number of entries in the following arrays.
unsigned_long*	pulTenderKeys	List of the tender key IDs.
const_wchar_t*	ppszNames	List of the tender key names.

## EQFASACCOUNTINQUIRYPARAMS

Contains all input and output parameters of getting information about a Campus Card account.

This structure is used by the following methods:

- eqFAS\_AccountInquiry

### Members of this structure

Type	Name	Description
unsigned_long	ulNumMapEntries	Size of structure (used for versioning).
unsigned_long	pulTenderKeys	Unique request ID (generated sequence number). The same value should be returned to ensure communication integrity.
const_wchar_t*	ppszNames	Primary identifier (that is, card number) of the Campus Card account coming from the DCE interface.
const_wchar_t*	pszSecondaryIdentifier	Secondary identifier (that is, password) of the Campus Card account coming from the DCE interface. It can be NULL or empty.
unsigned_long	ulTenderKey	The ID of the tender key whose information was requested.

Type	Name	Description
EQFASAPIEntryMethod	iEntryMethod	The authentication mode used when the primary and optionally the secondary identifiers were provided.
__int64	i64TransactionDate	The time of the request according to the caller.
const wchar_t*	pszCurrency	The currency symbol (in 3-letter ISO 4217 format) used by the caller. The returning current and minimum balance should represent the money in this unit. Conversion between currencies is the responsibility of the Campus Card integration.
double	dCurrentBalance	Output parameter containing the current balance of the Campus Card account. The value is in the provided currency
double	dMinimumBalance	Output parameter containing the minimum balance of the Campus Card account. DCE interface only allows such operations that do not decrease the current balance below the minimum. If it is 0, the total Campus Card account balance can be spent. If it is a negative number, the Campus Card account has a credit line. The value is in the provided currency.
wchar_t	aszNetworkUserId[256]	Output parameter containing an optional network ID (can be empty) of the Campus Card account. This value can be a Windows account name or an Equitrac account User ID. It is used to assign a Windows or Equitrac account to the Campus Card account to identify the available print jobs from the print queue.
wchar_t	aszResponseMessage[256]	Output parameter containing an optional response message (can be empty) logged by Equitrac.

## EQFASTRANSACTIONPARAMS

Contains all input and output parameters of a transaction.

This structure is used by the following methods:

- eqFAS\_DebitAccount
- eqFAS\_RefundAccount
- eqFAS\_DepositAccount

### Members of this structure

Type	Name	Description
unsigned long	cbSize	Size of structure (used for versioning).
unsigned long	ulSequenceNumber	Unique ID of the transaction (generated sequence number). The same value must be returned to ensure the communication integrity.

Type	Name	Description
const_wchar_t*	pszPrimaryIdentifier	Primary Identifier (that is, card number) of the Campus Card account coming from the DCE interface.
const_wchar_t*	pszSecondaryIdentifier	Secondary identifier (that is, password) of the Campus Card account coming from the DCE interface. It can be NULL or empty.
unsigned long	ulTenderKey	The tender key ID used for the transaction.
EQFASAPIEntryMethod	iEntryMethod	The authentication mode used when the primary and optionally the secondary identifiers were provided.
__int64	i64TransactionDate	The time of the request according to the caller.
const_wchar_t*	pszTransactionComment	Optional comment for the transaction (can be NULL or empty).
const_wchar_t*	pszCurrency;	The currency symbol used for the transaction (in 3letter ISO 4217 format). The received transaction amount and the returning current and minimum balance should represent the money in this unit. Conversion between currencies is the responsibility of the Campus Card integration.
double	dTransactionAmount	Amount to be deducted/refunded/deposited from/to the Campus Card account. The value is in the provided currency.
double	dCurrentBalance	Output parameter containing the current balance of the Campus Card account (after the transaction). The value is in the provided currency.
double	dMinimumBalance	Output parameter containing the minimum balance of the Campus Card account. DCE interface only allows such operations that do not decrease the current balance below the minimum. If it is 0, the total Campus Card account balance can be spent. If it is a negative number, the Campus Card account has a credit line. The value is in the provided currency.
wchar_t	waszResponseMessage[256]	Output parameter containing an optional response message (can be empty) logged by Equitrac.

## Methods

### eqFAS\_InitializeDLL

Initializes the Campus Card integration. Some global parameters can be initialized here

Called by System Manager and CAS immediately after the integrator DLL is loaded into memory.

### Parameters (in order)

Type	Name	Description
BOOL*	pbTracing	Pointer to a variable containing enablement of tracing. Note that the content of the variable can change between API calls, therefore it should be checked before each tracing. Tracing can be controlled via System Manager.
const wchar_t*	pszTraceFile	Full path of the trace file to be used, in case tracing is enabled. The trace file is provided by System Manager or CAS, and adding the trace messages to this file allows the System Manager to collect the trace file similar to other trace files.
const_wchar_t*	pszLanguageCode	The language System Manager and CAS use is displayed in a 2-letter format. It is recommended to use the same language for the configuration UI as was received.  Available values: "en", "de", "fr", "it", "pt" or "es" for English, German, French, Italian, Portuguese or Spanish, respectively.

### Return value

Type	Description
EQFASAPIRetCode	Result of the initialization, eFAS_OK if everything is OK. If initialization is unsuccessful, System Manager and CAS do not use the Campus Card system anymore and unload the DLL.

## eqFAS\_TerminateDLL

Uninitializes the Campus Card integration. Here some global parameters can be uninitialized and the memory can be freed.

Called by System Manager and CAS before the integrator DLL is loaded from memory and initialization was successful.

This method has no parameters.

### Return value

Type	Description
EQFASAPIRetCode	Result of the initialization, eFAS_OK if everything is OK. Failing the uninitialization does not have effect, the DLL is unloaded by Equitrac as normal. Only the trace file of Equitrac contains the failure information.

## eqFAS\_GetInfo

Returns information about the Campus Card integration.

Called by System Manager, when the Campus Card integration is selected on the Campus card interface dialog, and the returned information is displayed on the same dialog.

### Parameters (in order)

Type	Name	Description
unsigned int	uiInfoType	Reserved for future use, currently always 0.

### Return value

Type	Description
EQFASAPIINFO*	A pointer to a filled structure containing the information of the Campus Card integration. If there is an error, NULL should be returned. In this case System Manager does not use the Campus Card system anymore and unloads the DLL. Destroy the structure by calling the eqFAS_FreeInfo method.

## eqFAS\_FreeInfo

Frees the memory allocated by the eqFAS\_GetInfo method. Called by System Manager immediately after the eqFAS\_GetInfo method is called.

### Parameters (in order)

Type	Name	Description
unsigned int	uiInfoType	Reserved for future use, currently always 0.
EQFASAPIINFO	pInfo	Pointer to the structure that should be freed. Normally the same pointer was returned earlier by the eqFAS_GetInfo method.

### Return value

Type	Description
EQFASAPIRetCode	Result of the method, eFAS_OK if everything is OK. Failing the method does not have effect, the DLL is uninitialized and unloaded by System Manager as normal. Only the trace file of System Manager contains the failure information.

## eqFAS\_SetConfiguration

Receives and uses the Campus Card configuration from Equitrac.

**i** Although content and parsing the configuration is the responsibility of the Campus Card Integrator DLL, Equitrac stores the serialized configuration string in its database to ease integration.

Called by System Manager before the configuration UI (eqFAS\_ShowConfigurationUI) is requested. The UI controls can be set according to the received configuration.

Also called by CAS after the Campus Card Integrator DLL is loaded into memory and is initialized (eqFAS\_InitializeDLL). Connection, account information and transaction methods can use the saved configuration.

#### Parameters (in order)

Type	Name	Description
const wchar_t*	pszConfiguration	Configuration string (stored by Equitrac) to parse and process. It can be NULL or empty (if the Campus Card system is used with Equitrac the first time); in this case, a default configuration should be used.

#### Return value

Type	Description
EQFASAPIRetCode	Result of the method, eFAS_OK if everything is OK. Failing the method does not have effect, both System Manager and CAS use the DLL further as usual. Only the trace file of Equitrac contains the failure information.

## eqFAS\_GetConfiguration

Returns the currently used Campus Card configuration.

**i** Although content and parsing of the configuration is the responsibility of the Campus Card Integrator DLL, Equitrac stores the serialized configuration in its database to ease integration.

Called by System Manager after the configuration UI (eqFAS\_ShowConfigurationUI) is closed and returned with success. Therefore Equitrac can store the new configuration.

**i** If eqFAS\_SetConfiguration was not called before or was called but with an empty string, this method should return a default configuration.

This method has no parameters.

**Return value**

Type	Description
wchar_t*	A pointer to the serialized configuration of the Campus Card integration. If there is an error, NULL should be returned. The pointer can be freed when the eqFAS_FreeConfiguration method is called.

## eqFAS\_FreeConfiguration

Frees the memory allocated by the eqFAS\_GetConfiguration method. Called by System Manager immediately after the eqFAS\_GetConfiguration method is called.

**Parameters (in order)**


Type	Name	Description
const wchar_t*	pszConfiguration	Pointer to the string that should be freed. Normally the same pointer was returned earlier by the eqFAS_GetConfiguration method.

**Return value**

Type	Description
EQFASAPIRetCode	Result of the method, eFAS_OK if everything is OK. Failing the method does not have effect, the DLL is uninitialized and unloaded by System Manager as normal. Only the trace file of System Manager contains the failure information.

## eqFAS\_SetTenderKeyMap

Receives the allowed tender keys from Equitrac and uses them.

 A Campus Card system may have different tender keys, but not all of them must be used with Equitrac. The allowed tender keys can be selected on the configuration UI. Equitrac stores the allowed tender keys in its database to enhance integration.

Called by System Manager before the configuration UI (eqFAS\_ShowConfigurationUI) is requested. Allowed tender keys can be listed on the UI.

Also called by CAS after the Campus Card Integrator DLL is loaded into the memory and initialized (eqFAS\_InitializeDLL). Account information and transaction methods can have the list of the allowed tender keys.

**Parameters (in order)**

Type	Name	Description
EQFASAPITENDERKEYMAP	pTenderKeyMap	List of allowed tender keys (stored by Equitrac). It can be an empty list, if the Campus Card system is used with Equitrac the first time.

**Return value**

Type	Description
EQFASAPIRetCode	Result of the method, <code>eFAS_OK</code> if everything is OK. Failing the method does not have effect, both System Manager and CAS use the DLL further as normal. Only the trace file of Equitrac may contain the failure information.

## eqFAS\_GetTenderKeyMap

Returns the currently allowed tender keys.

**i** A Campus Card system may have different tender keys, but not all of them must be used with Equitrac. The allowed tender keys can be selected on the configuration UI. Equitrac stores the allowed tender keys in its database to ease integration.

Called by System Manager after the configuration UI (`eqFAS_ShowConfigurationUI`) is closed and returned with success. Therefore Equitrac can store the selected tender keys.

This method has no parameters.

**Return value**

Type	Description
EQFASAPITENDERKEYMAP	A pointer to the list of configured tender keys. If the Campus Card system does not support tender keys, the returned list should contain at least one item. Otherwise Equitrac automatically uses a default tender key with ID "1" and name "Default". The name of the returned tender keys become visible on the account selection screen of the DCE interfaces. If there is an error, NULL should be returned. The pointer can be freed when the <code>eqFAS_FreeConfiguration</code> method is called.

## eqFAS\_FreeTenderKeyMap

Frees the memory allocated by the `eqFAS_GetTenderKeyMap` method. Called by System Manager immediately after the `eqFAS_GetTenderKeyMap` method is called.

**Parameters (in order)**

Type	Name	Description
EQFASAPITENDERKEYMAP	<code>pTenderKeyMap</code>	Pointer to the list of tender keys that should be freed. Normally the same pointer was returned earlier by the <code>eqFAS_GetTenderKeyMap</code> method.



**Return value**

Type	Description
EQFASAPIRetCode	Result of the method, <code>eFAS_OK</code> if everything is OK. Failing the method does not have effect, the DLL is uninitialized and unloaded by System Manager as normal. Only the trace file of System Manager contains the failure information.

## eqFAS\_ShowConfigurationUI

Shows the configuration dialog containing controls to configure the Campus Card integration. The controls should be initialized based on the current configuration and allowed tender keys.

Called by System Manager after the saved configuration and tender keys are sent to the Integrator DLL (`eqFAS_SetConfiguration`, `eqFAS_SetTenderKeyMap`).

It is recommended to show the localized version of the configuration dialog based on the received language code at the initialization (`eqFAS_InitializeDLL`).

Ensure that the configuration dialog is a modal dialog, as System Manager performs less than optimal while the configuration dialog is open.

If the integrated Campus Card system supports tender keys, a table for selecting the allowed tender keys may be required on the configuration UI. In this case the tender key grid control of the `FASampleWithUI` sample application can be used (see [Sample projects](#)). Additionally, the list of the selected tender keys cannot be empty. Otherwise Equitrac automatically uses a default tender key with ID "1" and name "Default", which may always cause authentication problems if the integrated Campus Card system does not have a tender key with such data.

**Parameters (in order)**

Type	Name	Description
HWND	<code>hwndParent</code>	Handle of the parent window (which is normally the System Manager).

**Return value**

Type	Description
EQFASAPIRetCode	Result of the method. IF <code>eFAS_OK</code> is returned, System Manager gets the new configuration and tender keys to save them ( <code>eqFAS_GetConfiguration</code> , <code>eqFAS_GetTenderKeyMap</code> ).

## eqFAS\_Connect

Establishes the communication channel with the Campus Card system.

Called by CAS after the DLL is loaded into memory, initialized (`eqFAS_InitializeDLL`), then configurations and tender keys are set (`eqFAS_SetConfiguration`, `eqFAS_SetTenderKeyMap`).

When establishing the connection is slow, this method is the right place for it. The account information and payment transaction methods should finish in 10 seconds because of the time guard in CAS (as described in [Implementing the DLL](#)); there is no time for those methods to establish the connection. For example, a background thread can be started at this point to create and maintain the connection. However, if connecting with the Campus Card system is fast and complicated threading is not feasible, this method may be pointless.

This method has no parameters.

#### Return value

Type	Description
EQFASAPIRetCode	Result of the method, <code>eFAS_OK</code> if everything is OK. If the method fails, CAS does not use the DLL further, but terminates and unloads it.

## eqFAS\_Disconnect

Shuts down the communication channel to the Campus Card system. It should do the opposite of what the connection request (`eqFAS_Connect`) did.

Called by CAS before the Integrator DLL is unloaded.

This method has no parameters.

#### Return value

Type	Description
EQFASAPIRetCode	Result of the method, <code>eFAS_OK</code> if everything is OK. Failing the method does not have effect, the DLL is terminated and unloaded by CAS as normal. Only the trace file of CAS contains the failure information.

## eqFAS\_AccountInquiry

Tries to find a Campus Card account and returns its properties. A Campus Card account can be found based on the provided identification data (primary and optionally secondary identifier and optionally a tender key ID). If an account can be found, its current and minimum balances, and optionally its network ID should be returned.

Called by CAS, if a login request is coming from a DCE interface.

**i** The method finishes in 10 seconds because of the time guard in CAS (see [Implementing the DLL](#)).

**Parameters (in order)**

Type	Name	Description
const EQFASACCOUNTINQUIRYPAR AMS*	pAccountInquiryParams	Pointer to the structure containing the identification data, and the place where the returned Campus Card account information must be populated.


**Return value**

Type	Description
EQFASAPIRetCode	Result of the method: eFAS_OK if an account was found and returned. eFAS_SecondaryPINRequired if an account was found, but not returned because a password is also required and the provided secondary identifier is empty. If the method fails, CAS behaves as if no Campus Card account is found.

## eqFAS\_DebitAccount

Debits a Campus Card account. A Campus Card account can be found based on the provided identification data (primary and optionally secondary identifier and optionally a tender key ID). If an account can be found, its balance should be decreased with the specified amount. Finally, the new balance is returned.

Called by CAS, if the DCE interface wants to charge the logged-in Campus Card account as a result of an SFD/MFD operation.

 The method finishes in 10 seconds because of the time guard in CAS (see [Implementing the DLL](#)).

**Parameters (in order)**

Type	Name	Description
const EQFASTRANSACTIONPARAMS	pTransactionParams	Pointer to the structure containing the identification data and the transaction amount, and the place where the returned new balance must be populated.

**Return value**

Type	Description
EQFASAPIRetCode	Result of the method: eFAS_OK if an account was found and returned. eFAS_SecondaryPINRequired if an account was found, but has not enough money. If the method fails, the DCE interface does not allow the operation.

## eqFAS\_RefundAccount

Refunds a Campus Card account. A Campus Card account can be found based on the provided identification data (primary and optionally secondary identifier and optionally a tender key ID). If an account can be found, its balance should be increased with the specified amount. Finally, the new balance is returned.

Currently not used by Equitrac, but a future version of Equitrac might use this.

**i** The method finishes in 10 seconds because of the time guard in CAS (see [Implementing the DLL](#)).

### Parameters (in order)

Type	Name	Description
const EQFASTTRANSACTIONPARAMS	pTransactionParams	Pointer to the structure containing the identification data and the transaction amount, and the place where the returned new balance must be populated.

### Return value

Type	Description
EQFASAPIRetCode	Result of the method: eFAS_OK if an account was found and charged.

## eqFAS\_DepositAccount

Deposits a Campus Card account. A Campus Card account can be found based on the provided identification data (primary and optionally secondary identifier and optionally a tender key ID). If an account can be found, its balance should be increased with the specified amount. Finally, the new balance is returned.

Currently not used by Equitrac, but a future version of Equitrac might use this.

**i** The method finishes in 10 seconds because of the time guard in CAS (see [Implementing the DLL](#)).

### Parameters (in order)

Type	Name	Description
const EQFASTTRANSACTIONPARAMS	pTransactionParams	Pointer to the structure containing the identification data and the transaction amount, and the place where the returned new balance must be populated.

**Return value**

<b>Type</b>	<b>Description</b>
EQFASAPIRetCode	Result of the method: eFAS_OK if an account was found and charged.

# Test application

## Prerequisites

The Campus Card Test Application is a tool for easily testing Campus Card Integrator DLLs via a graphical user interface. It can be used for calling each entry point of the Campus Card API, used by Equitrac.

For running the test application, the Microsoft Visual C++ 2010 SP1 Redistributable Package (32-bit or 64-bit version) must be installed. These can be downloaded from the following links:

- <http://www.microsoft.com/en-us/download/details.aspx?id=8328>
- <http://www.microsoft.com/en-us/download/details.aspx?id=13523>

**i** If Microsoft Visual Studio 2010 SP1 is installed, the required C++ runtimes are already installed with it.

## Start the test application

1. To run the test application, navigate to the appropriate folder (TestApp\32 or TestApp\64) of the API package and launch the `FASPluginTester.exe` file (same file name for both the 32-bit and the 64-bit editions). The FAS Plugin Tester dialog appears with several controls and settings (described in the next section).
2. To ease test application usage, the recently used settings are saved into a separate `FASPluginTester.xml` file next to the executable. Once the test application is started again, the recently used settings are automatically populated in the dialog; therefore they need not be set again. For this comfortable functionality the test application should start from a folder, for which write access is granted.

## Use the test application

The test application window can be seen on the picture below.

1. Select a Campus Card Integrator DLL to be tested. Enter the full path of the DLL into the **Plugin DLL** textbox, or press the **Browse** button to open Windows' built-in open file dialog and select the DLL.

2. Press the **Load** button under the **Plugin DLL** textbox to load the DLL into memory.
3. Once the DLL file is loaded, the **Unload** button becomes active. Press it to unload the DLL from the test application.
4. If the DLL file is loaded, the **Initialize** group box becomes active.
5. Initialization settings (eqFAS\_InitializeDLL method parameters) can be specified here. These are the following:
  - **Language code** ('en' for English, default setting).
  - **Enable tracing** check box to switch on tracing.
  - **S** field to specify the full path of the trace file to be used.
6. Once the DLL is initialized, the following further areas become active:
  - **UninitializeDLL** button (1)
  - **Plugin information** group box (2)
  - **Tender keys** group box (3)
  - **Custom, -per DLL configuration** group box (4)
  - **Connect** button (5)
7. Press the **UninitializeDLL** button (1) to uninitialized the DLL by calling its eqFAS\_TerminateDLL method.
8. Press the **Get plugin info** button (2) to get information about the Campus Card Integrator DLL by calling its eqFAS\_GetInfo and eqFAS\_FreeInfo methods and populate the **Short name** and **Description** fields with the returned data.

The image above shows the returned data of the FASSampleWithUI sample DLL (See [Sample projects](#)).
9. The allowed tender keys can be edited in the **Tender keys** group box.

**i** Supporting Tender Keys are not mandatory, it is an optional feature to allow several purses for an account. Any further settings about the tender keys can be implemented in the Campus Card Integrator DLL's own configuration UI (see later).

10. Press the **Get** button (2) to acquire the allowed tender keys of the Campus Card integration by calling the eqFAS\_GetTenderKeyMap and eqFAS\_FreeTenderKeyMap methods and show them in the tender keys table (1).
11. Tender keys can be added or removed by right-clicking the tender keys table (1) and selecting the **Add** and **Remove** items from the context menu, respectively.

**i** The **Remove** button is only active when right-clicking on an already existing tender key.

12. Press the **Set** button (3) to send the currently allowed tender keys in the tender key grid to the integrator DLL by calling its eqFAS\_SetTenderKeyMap method.
13. The custom configuration settings of the Campus Card Integrator DLL can be managed through the Custom, perDLL configuration group box.
14. Press the **Show configuration** UI button (1) to open the custom configuration dialog of the integrator DLL by calling the eqFAS\_ShowConfigurationUI method. If the method is returned with success (eFAS\_OK, see [EQFASAPIRetCode](#)), the allowed tender keys and the new configurations settings are acquired from the integrator DLL (using its eqFAS\_GetTenderKeyMap, eqFAS\_FreeTenderKeyMap, eqFAS\_GetConfiguration and

eqFAS\_FreeConfiguration methods) and populate the **Tender keys** table and the **Configuration string** textbox (3), respectively.


15. Press the **Get** button (2) to obtain the current configuration settings of the integrator DLL by calling its eqFAS\_GetConfiguration and eqFAS\_FreeConfiguration methods. The acquired, serialized configuration string is displayed in the **Configuration string** textbox (3).
16. The recently acquired, serialized configuration string is displayed in the **Configuration string** textbox (3). By default, the configuration string is displayed as returned by the integrator DLL, usually a long string. To review it easier, word wrapping can be switched on with the **Wrap** check box (4). For more details, press the '...' button (5) to show the configuration string in a new, bigger dialog. The configuration string can be also edited here or in the **Configuration string** textbox (3).
17. Press the **Set** button (6) to send the configuration string currently displayed in the **Configuration string** textbox (3) to the integrator DLL by calling its eqFAS\_SetConfiguration method.
18. Press the **Connect** button to tell the Campus Card Integrator DLL that connection with the Campus Card system can be established if required, by calling its eqFAS\_Connect method. If the connection is successful, the following areas become active:
  - **Disconnect** button (1)
  - **Transaction handling** group box (2)
19. Press the **Disconnect** button (1) to tell the integrator DLL to terminate connection established earlier, by calling its eqFAS\_Disconnect method.
20. In the **Transaction handling** group box (2) Campus Card operations can be initiated and their results are displayed.
21. Select the operation from the **Transaction** drop-down list (3); there are four types of operation:
  - **Account inquiry** to get information about a Campus Card account (by calling the eqFAS\_AccountInquiry method of the integrator DLL).
  - **Debit account** to debit a Campus Card account (by calling the eqFAS\_DebitAccount method of the integrator DLL).
  - **Deposit account** to deposit a Campus Card account (by calling the eqFAS\_DepositAccount method of the integrator DLL).
  - **Refund account** to refund a Campus Card account (by calling the eqFAS\_RefundAccount method of the integrator DLL).

The parameters of the selected operation should be specified with the controls below the **Transaction** drop-down list (2) (see the EQFASACCOUNTINQUIRYPARAMS and EQFASTRANSACTIONPARAMS structures).

22. Specify the primary and optionally the secondary identifiers of the used Campus Card account in the **Primary ID** (4) and **Secondary ID** (5) fields, respectively.
23. Select or type the used currency format in the **Currency** drop-down list (6) in a 3-letter ISO 4217 format.
24. The **Trans. date** fields (7) (transaction date) provide a means for setting the exact date and time for the operation. It is also possible to specify the current date and time on an as-of-now basis to the precision of one second. This results in an updated date and time every second. Press the **Set to Now** button (8) (repeatedly) to achieve this.



25. If the selected operation is **Debit account**, **Deposit account**, or **Refund account**, specify the amount of the transaction in the **Trans. amount** field (9). Additionally a comment can be set for the transaction in the **Trans. comment** field (10).
26. Select the emulated authentication mode in the **Entry method** group box (11) as Card swipe or Keypad entry.
27. Provide a tender key (if tender keys are supported and used) for the operation in the **Tender key** field (12). Optionally, override the default sequence number in the **Sequence no.** field (13).
28. If all the parameters are provided, press the **>>Send transaction<<** button (14) to perform the selected operation.

 If the operation timeouts (recall the time guard of Equitrac described in [Implementing the DLL](#)), a warning message is displayed.

29. When the operation is completed, its results are displayed below the **>>Send transaction<<** button (14).