VETtrak API

Introduction

An API into the VETtrak software and database has been produced which allows web services to interact with the VETtrak database.

This document is intended for Learning Management Systems (LMS) providers, online portal providers and others who may wish to integrate VETtrak data and/or functionality into their web-based applications.

Comments and feedback are encouraged to allow VETtrak Pty Ltd to further develop the API in conjunction with those who may wish to use it in their own development projects.

Whilst this document is aimed mainly at the LMS provider sector, there are a number of other areas for integration that will also be available within the VETtrak API.

VETtrak API Overview

VETtrak API is a software product that allows the VETtrak training management system to be linked with other web-based software systems and products, so that data can be transferred between the participating systems.

VETtrak API can be used by:

- Existing VETtrak users who want to use their VETtrak data in other business systems.
- Businesses that have training data in other systems and want to use VETtrak's management and reporting capabilities on that data.

The main types of applications that can be integrated include:

- Learning Management Systems (LMS)
- Online Enrolment Systems
- Portals which allow various parties to view training information online

There are three separate products, one addressing each of these application types. Since each product uses the same methodology to address different functionality, the term API (or VETtrak API) will be used to refer to any of them.

Technical

The VETtrak API is provided via a comprehensive suite of SOAP-based web services. Such web services can be consumed easily by many different web-based languages, including PHP or .Net.

The API is provided in the form of dynamic link library (dll) files. There is one dll file for each product.

The VETtrak API must reside on a Windows-based web server running IIS 5 or better. The API is not provided for other platforms.

The VETtrak API connects to the VETtrak database and transfers data back and forth as required. The API and the database must both reside on the same physical network.

Implementation Scenario

The API resides in a virtual directory of a web server on the same network as the VETtrak database. Usually this would be the Registered Training Organisation's (RTO's) internal network.

If not already available, a web server will need to be installed on the RTO's internal network to host the API.

The RTO needs to look after security of their network, firewalling, and 24/7 access to their web server.

The web application that runs the RTO's website will need to be developed to make use of the functionality of the API.

VETtrak Terms

As the intended audience of this document may not be actual users of VETtrak, some explanation of terms is given for the following:

General	
Client	An individual person, usually a student, learner or trainee. Each client has a unique identifier (10 char string) called the client code.
Staff member	A client who has been flagged as being a staff member
Employer	A company, business or organisation
Qualification	A programme of study that leads to a nationally recognised certification. A Qualification is made up of units from Training Packages

Course	A programme of study that is not nationally recognised. This may be state accredited or it may be specific to a particular training organisation.
Unit	A component of a Qualification. An assessment is made against a Unit.
Module	A component of a Course. An assessment is made against a Module.
AVETMISS	Australian Vocational Education and Training Management Information Statistical Standard. The agreed national data standard for the collection, analysis and reporting of vocational education and training information in Australia.
Short Course Based	
Programme	A set of education or training activities designed to achieve a specific outcome. Example: Beer Appreciation
Occurrence	An instance of a Programme. Each occurrence has a unique identifier called an Occurrence ID. The Occurrence can hold AVETMISS data and other information. This information becomes a template for subsequent enrolments into the Occurrence. For example, an Occurrence may have: • a list of Units • a list of Classes • a list of Prices • a number of vacancies • a location (where offered) • a start and end date (when offered) • a link to a qualification or course Example: An offering of the Beer Appreciation programme which includes: • 2 Units: o How to serve beer o Different styles of beer

	• Commence Date: Sept 1, 2008
	• Finish Date: Sept 27, 2008
	• Location: Melbourne Campus
	• 4 Classes
	  o Monday, 1 Sept 2008, 6pm – 9pm, Room A1
	  o Tuesday, 9 Sept 2008, 6pm – 9pm, Room B21
	  o Friday, 20 Sept 2008, 1pm – 5pm, Room C321
	  o Friday, 27 Sept 2008, 4pm – 6pm, Room A1
	• 10 Seats Available
	• Prices:
	  o Pensioners $50
	  o Children $20
	  o Full Fee $100
	• Part of Certificate 2 in Bar Tending

Occurrence Enrolment	Registration of a Client to participate in an Occurrence. Each enrolment has a unique identifier called the enrolment ID. The enrolment takes on all the features of the Occurrence but may be adjusted if required. As part of the enrolment process the client is enrolled into any Units that are part of the Occurrence. Example: John Smith enrols in the Beer Appreciation Occurrence mentioned above. This means John Smith is also enrolled in each of the units: • How to serve beer • Different styles of beer
Trainee	A person undertaking a Traineeship or New Apprenticeship. This is a system of vocational training combining off-the-job training at an approved training provider with on-the-job training and practical work experience.
Contract	An agreement between a trainee and an employer that outlines the training to be undertaken. A contract must be related to a specific qualification or course.
Class	A meeting at which training is undertaken.
Event	Something that happens to a person or employer at a particular time, or over a period of time. This is

	commonly used to record workplace visits to trainees, but can also be used to record annual leave, correspondence or other events.
Programme Type	A category which the various programmes are divided into. These are set up according to the needs of the particular training organisation and do not conform to any standard.
Training Provider	An organisation that delivers training of some sort.
RTO	Registered Training Organisation. A Training Provider that is registered to deliver nationally recognised training (part of a training package). Only RTOs can issue nationally recognised qualifications.
LMS	Learning Management System. A web-based software package that enables the delivery and management of training to students.

Broad Functional Areas of API

LMS Integration – Integrate API

These functions are used to support the transfer of information between an LMS provider and VETtrak. This topic is discussed in more detail later in this document.

The aim is twofold: to allow the training activity under the LMS to be available to the VETtrak reporting and management functions, and to allow the LMS to import existing VETtrak data (such as student records).

On-Line Enrolment – Enrol API

This allows a client or employer to enrol in an Occurrence online over the web. VETtrak Pty Ltd has an existing product in this space that is intended to be integrated into the API and the existing product eventually dropped.

Portals – Portals API

All of the portals described below allow viewing of data only; no changes can be made to data via a portal, other than to the user's own details.

Employer Portal

This allows an Employer to access, via the web, various training information concerning people related to the employer's business.

Client Portal

This allows a Client to access, via the web, various training information concerning him/herself.

Staff Portal

Access to the Staff portal includes access to the client portal. In addition, the staff member has access to other relevant information.

Sample portals are available which link to a demo database, see links on page 23.

General Implementation

The idea of how the API is implemented by a web application, in general terms, is as follows.

Authentication and Tokens

- A user attempts to authenticate by submitting a User Name and Password.
- On successful authentication, a string token is returned from the API to the web app.
- The token should be stored by the web app and used for future function calls to the API.
- When a token is successfully used, it is renewed - its 'age' is reset to zero. This allows a user's session to be extended indefinitely, so long as activity continues.
- A token has a limited lifetime, (which is configurable in the web app). It will expire if it is not renewed.
- If an expired token is passed with a function call, the function will fail and it will be necessary for the user to authenticate again.

Handling Tokens

The descriptions below use the functions *`ValidateClient`* and *`GetEventsForClient`* as examples. These functions have the following definitions:

```
      function ValidateClient(sUsername, sPassword: String):
TAuthenticate;
```
(see Figure 1 on page 8)

```
      function GetEventsForClient(sToken, sClie_Code: String):
TAuthEvenList;
```
(see Figure 2 on page 9.)

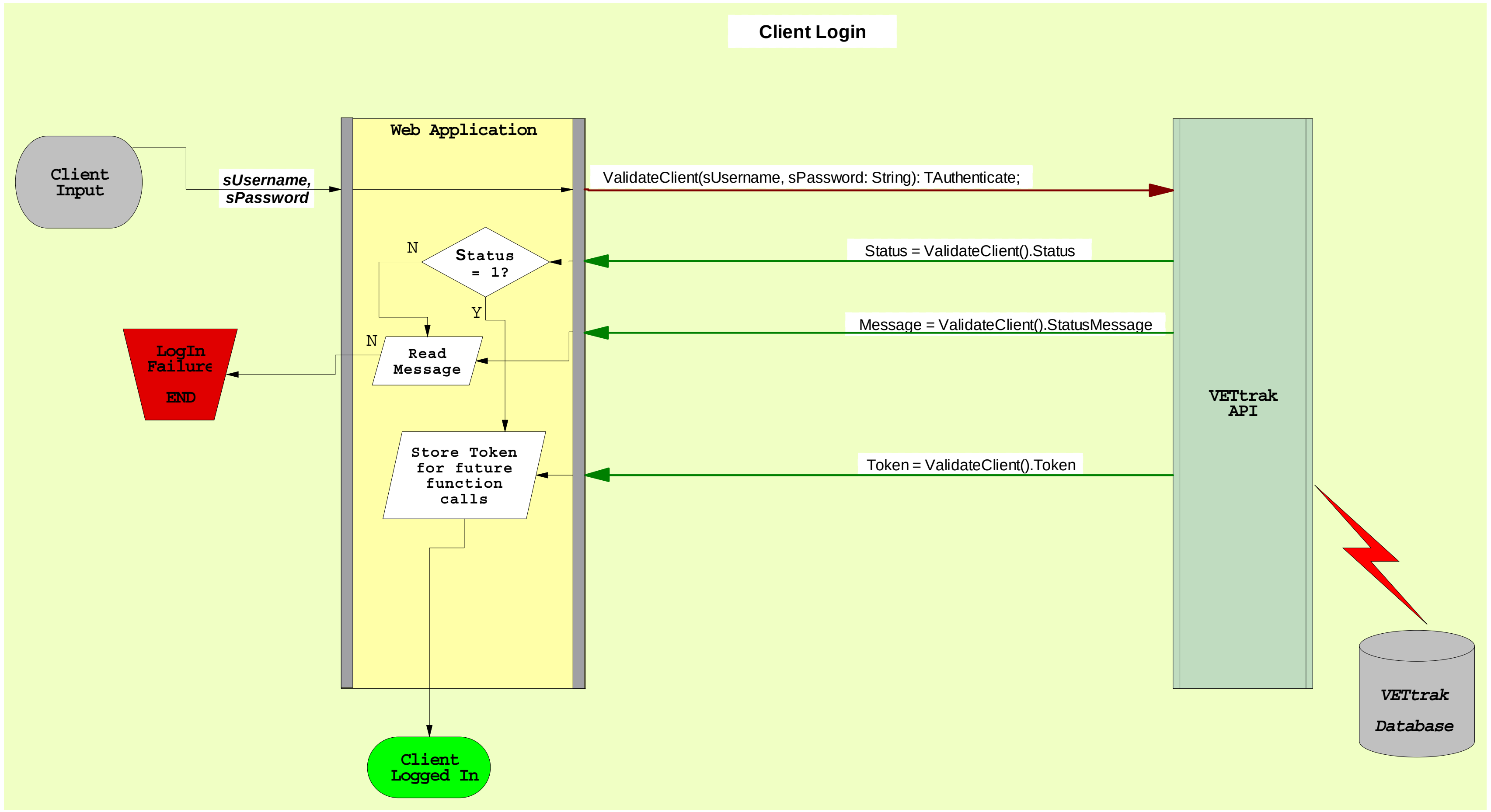

Client Login
Web Application
Client Input
sUsername, sPassword
ValidateClient(sUsername, sPassword: String): TAuthenticate;
Status = 1?
N
Y
Status = ValidateClient().Status
Message = ValidateClient().StatusMessage
N
Read Message
LogIn Failure END
Store Token for future function calls
Token = ValidateClient().Token
VETtrak API
VETtrak Database
Client Logged In

Figure 1: Message Flow of ValidateClient Function

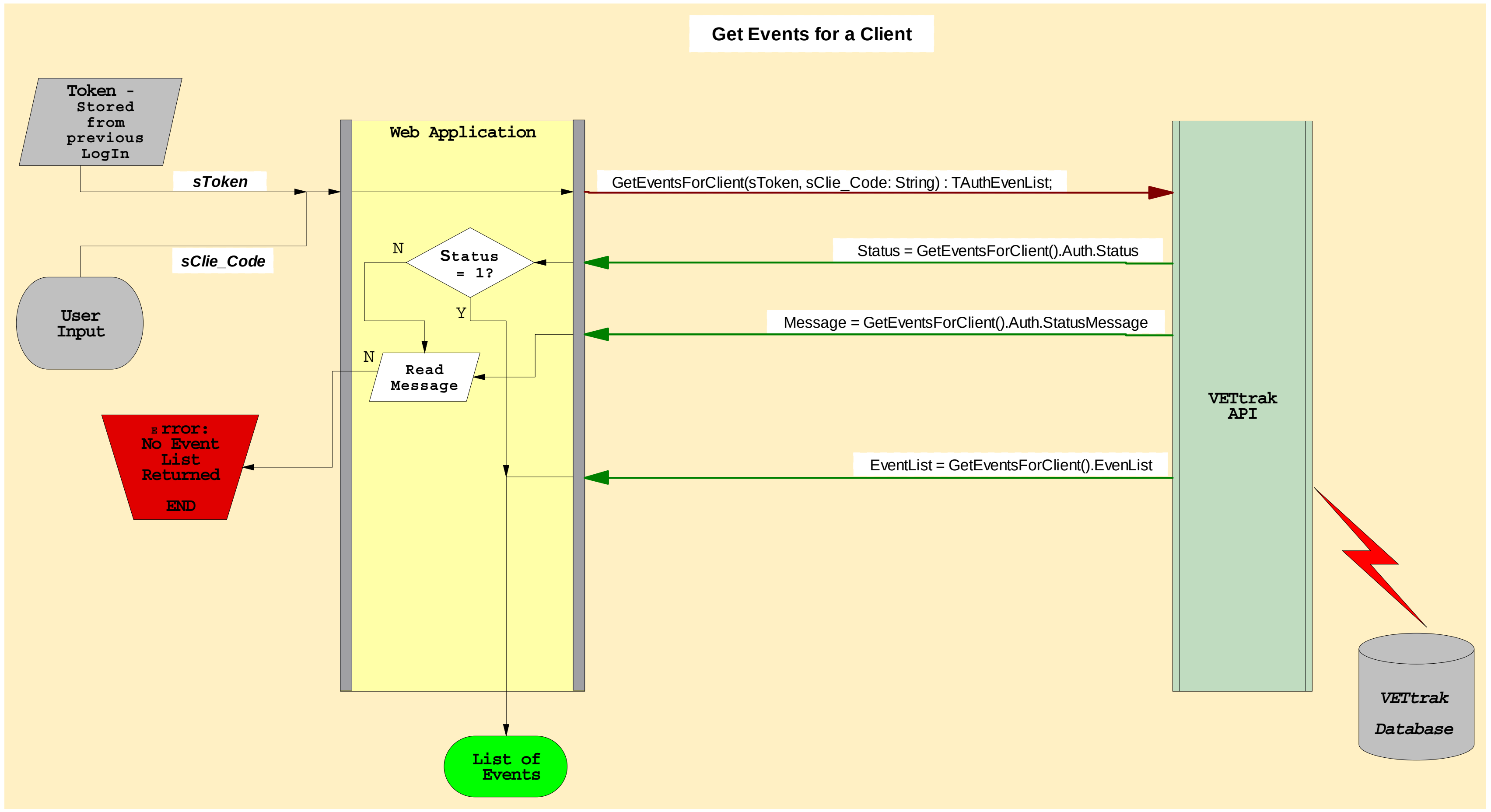

Get Events for a Client
Token - Stored from previous LogIn
sToken
sClie_Code
User Input
Web Application
GetEventsForClient(sToken, sClie_Code: String) : TAuthEvenList;
Status = GetEventsForClient().Auth.Status
Message = GetEventsForClient().Auth.StatusMessage
EventList = GetEventsForClient().EvenList
Status = 1?
N
Y
N
Read Message
Error: No Event List Returned END
VETtrak API
VETtrak Database
List of Events

Figure 2: Message Flow of GetEventsForClient Function

- ***Pass Token from API to the Web App***

A token is passed from the API back to the Web App as the ***`.Token`*** property of a `TAuthenticate` object. This `TAuthenticate` object will be a function result, or in some cases, a property (the ***`.Auth`*** property) of a function result.

In our examples, this translates to:

```
Token = ValidateClient(sUsername, sPassword).Token
```

and

```
Token = GetEventsForClient(sToken, sClie_Code).Auth.Token
```

Other properties of the `TAuthenticate` class describe characteristics of the Token. See the `TAuthenticate` Class documentation for more details. In particular, many functions will make use of the ***`Status`*** and ***`StatusMessage`*** properties.

- ***Pass Token from Web App to the API***

A token is passed to the API as a parameter to a function.

Example:

```
GetEventsForClient(sToken, sClie_Code)
```

All functions which potentially return any non-public domain data will require a Token as a parameter.

What about errors?

When calling an API function, there are several potential errors which can arise. Some situations which would produce errors are:

- An invalid or expired token is passed as a parameter.
- A bad parameter is passed - one which does not correspond to an appropriate record in the VETtrak database.
- In the `GetEventsForClient(sToken, sClie_Code)` example, the `sClie_Code` parameter might not represent a valid Client in the VETtrak database.
- There are problems connecting to the VETtrak database, for example, the database server is down.
- There is bad data in the database.

If an API function returns an error, this is communicated in the ***`Status`*** and ***`StatusMessage`*** properties of the `TAuthenticate` object. The ***`Status`*** property is an integer which provides a broad status category, while the *`StatusMessage`* property is a string which provides a message when necessary.

In our examples, we would have:

```
Status = ValidateClient(sUsername, sPassword).Status,
      StatusMessage = ValidateClient(sUsername, Password)
.StatusMessage
```

and

```
      Status = GetEventsForClient (sToken, sClie_Code).Auth.Status,

      StatusMessage = GetEventsForClient (sToken, sClie_Code)
.Auth.StatusMessage
```

Status Codes used in the API are:

```
      STATUS_NO_RECORDS = -4;
      STATUS_DB_ERROR   = -3;
      STATUS_EXPIRED    = -2;
      STATUS_ERROR      = -1;
      STATUS_UNKNOWN    =  0;
      STATUS_OK         =  1;
```

In most cases, the web app should check for `Status = 1.` This is the expected status when no errors occur.

Often, if no errors occur, the `statusMessage` is empty.

If errors occur, the properties of a `TAuthenticate` object do not provide any information about the Token. In particular, the `Token` property is empty.

LMS Integration Scenario

There are a number of scenarios that could be used to transfer information either from VETtrak to the LMS or vice versa. Some typical scenarios are described here.

Preparation for integration

For the integration to work, there needs to be some linkage set up between the courses offered on the LMS and the Occurrences held within VETtrak. VETtrak has a unique identifier that can be used to identify an Occurrence. This identifier, an integer, would be linked to a course on the LMS.

An alternative might be for the Occurrence to hold an identifier that uniquely identifies a course on the LMS.

All development to date has been based on the former scenario.

Occurrence set up in VETtrak and transferred to LMS

An Occurrence to be delivered by the LMS must be established in VETtrak, in the usual manner for creating a new occurrence. It should be flagged as an LMS Occurrence. Another flag, we will call it the *Uploaded* flag, indicates whether that occurrence has been implemented in the LMS yet.

The API provides a function: `GetLMSOccurrencesInDateRange()`.

This function returns a list of Occurrences that are flagged as LMS Occurrences. A date range can be passed as parameters to restrict the resulting occurrences to those that run within the requested range.

Another available function is: `GetLMSNewOccurrences()`.

This function returns those LMS Occurrences that are new to the LMS. This means LMS Occurrences with *Uploaded* flag set to ‘not yet’.

Either of these functions can be used to offer the LMS administrator a list of VETtrak Occurrences. The LMS administrator can then select the appropriate occurrence and capture its VETtrak Occurrence ID. This Occurrence ID can be used when the course offering is set up in the LMS, to link the LMS course with the VETtrak occurrence.

After the course is successfully set up in the LMS, the LMS administrator can inform VETtrak that the *Uploaded* flag should be set to ‘done’. This removes the VETtrak occurrence from the list returned by a call to `GetLMSNewOccurrences()`.

The API provides a function, `UpdateLMSOccurrenceStatus()`, for this purpose.

The VETtrak Occurrence ID is passed as a parameter.

Client Enrolled in VETtrak and transferred to LMS

Once the Occurrence is linked to a course offering in the LMS, the following process can occur:

- A Client is enrolled into an Occurrence. This may be a new or existing Client.
- The Occurrence Enrolment is flagged as "to be uploaded".
- An LMS process runs to collect new enrolments that have not yet been uploaded.
  (Call to function: `GetLMSNewEnrolmentsForOccurrence`)
- The LMS checks the client code, and either creates a new client or finds an existing one.
- The LMS enters the client into a course offering, as per normal procedures.
- The LMS informs VETtrak that the enrolment has been successfully transferred to the LMS and that the Occurrence Enrolment should now be flagged as "has been uploaded".
  (Call to function: `UpdateLMSEnrolmentStatus`)

Client entered in LMS and transferred to VETtrak

Once an Occurrence is linked to a course offering in the LMS, the following process can occur:

- A Client is enrolled into a course offering as per normal LMS procedures.
- Each client enrolment is sent to the API in turn.
- The API checks if the client exists in VETtrak.

  If the (suspected) VETtrak client code is known, call to function: `GetClientByCode` (see Figure 3 on page 15).

  If the code is unknown, call to function:
  `GetClientByName`

  In either case, a client code is returned if the client exists, otherwise the returned client code is an empty string ("").
- If the Client is new, a client code is created (allocated automatically by the API) and returned to the LMS to be stored for future calls.
  (Call to function: `AddClientAfterCheck`)
- The client is then enrolled into the given Occurrence.
- The LMS runs a process that transfers client enrolments in the LMS to the equivalent Occurrence Enrolments within VETtrak.
  (Call to function: `AddLMSClientEnrolmentToOccurrence`)
- Parameters are passed to identify the client's code and the occurrence ID.

- In both cases, (new or existing client), a return message is sent that contains the new enrolment ID.
- The LMS should record the client code and enrolment ID for later API calls.

Results recorded in LMS and transferred to VETtrak

Once a Client has been enrolled in a course offering, the following process can occur to transfer the results from the LMS to VETtrak.

- An LMS process runs to look for newly recorded results that are to be transferred.
- Each result is sent to the API in turn, via a call to the function: `UpdateResult`.
- For each result, the following information is sent to the API as parameters to the function, `UpdateResult`:
  - o the Unit/Module code to identify the Unit
  - o the enrolment ID to identify the enrolment
  - o the Result to identify the result
  - o the Start and End date of studying this Unit
  - o the Assessment Score (as a percentage)
- The given results are recorded for the Unit/Module in the given enrolment.
- Confirmation is sent back to the LMS.

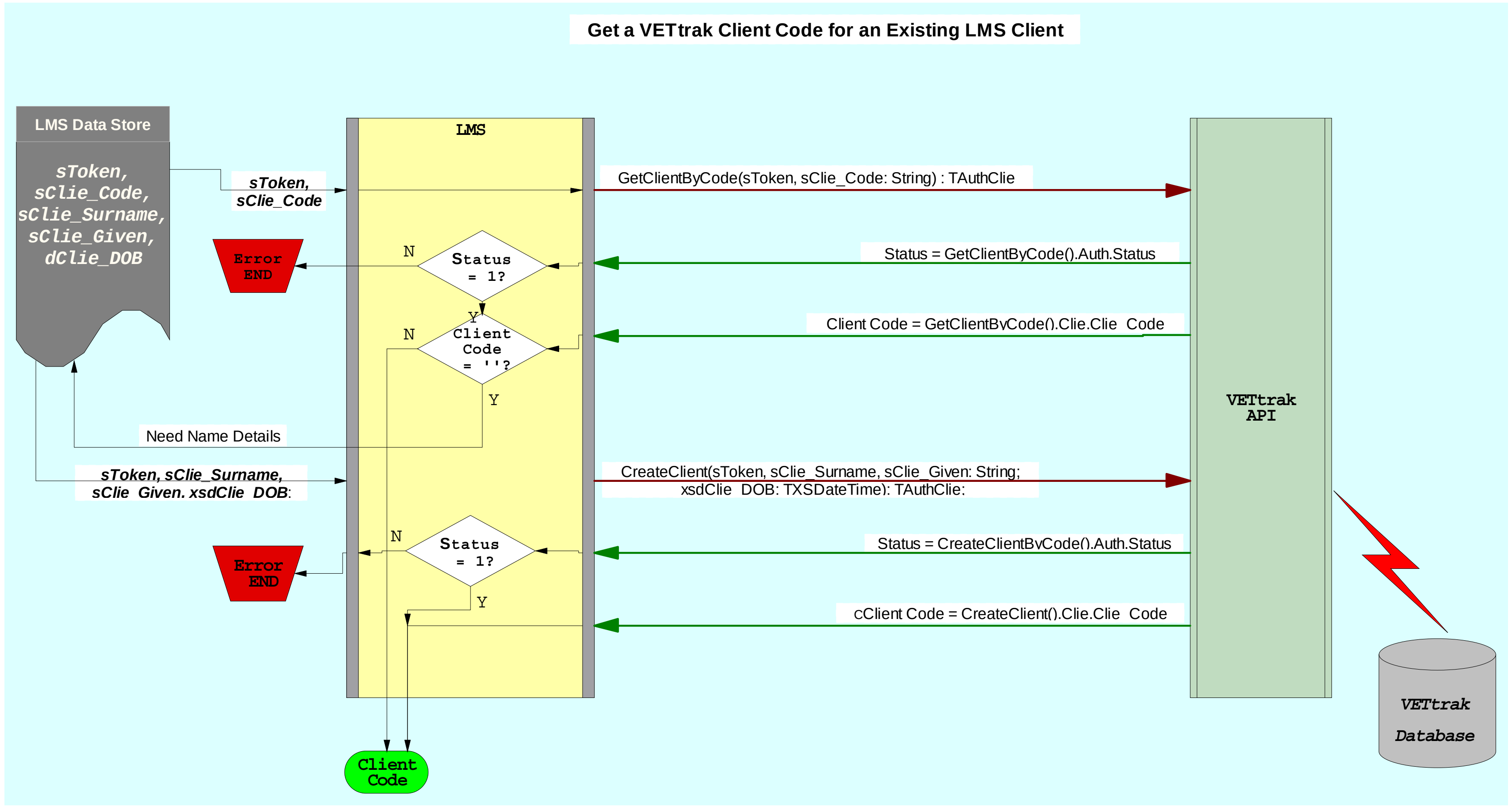

Get a VETtrak Client Code for an Existing LMS Client
LMS Data Store
sToken, sClie_Code, sClie_Surname, sClie_Given, dClie_DOB
sToken, sClie_Code
LMS
GetClientByCode(sToken, sClie_Code: String) : TAuthClie
Error END
N
Status = 1?
Status = GetClientByCode().Auth.Status
Y
N
Client Code = ''?
Client Code = GetClientByCode().Clie.Clie_Code
Y
Need Name Details
VETtrak API
sToken, sClie_Surname, sClie_Given. xsdClie_DOB:
CreateClient(sToken, sClie_Surname, sClie_Given: String; xsdClie_DOB: TXSDateTime): TAuthClie:
N
Status = 1?
Status = CreateClientByCode().Auth.Status
Error END
Y
cClient Code = CreateClient().Clie.Clie_Code
VETtrak Database
Client Code

Figure 3: Message Flow of GetClientByCode Function

Web Enrolment Scenario

The Web Enrolment API is designed to allow prospective students to enrol in training via a form on a website.

Such enrolment details need to be recorded in the VETtrak database, but are not subjected to the validation procedures provided by the VETtrak application. This gives rise to the possibility of bad or duplicate data entering the database.

The solution adopted is to store all of the web enrolment details in quarantined tables. The concepts of a ‘web client’, ‘web enrolment’, ‘web company’, ‘web employee’ and so on, are introduced to indicate entities that have been created via the web enrolment process without further validation.

In the event that these entities can be validated, and possibly indentified as existing entities in the VETtrak database, they become a regular ‘client’, ‘enrolment’, ‘employer’, etc.

The validation is done in an extension in the VETtrak application, which allows a staff member to attempt to match up the web enrolment clients, for example, with existing clients, before processing these 'web enrolments' into standard enrolments.

The VETtrak extension needed for the validation is included with the Enrol API product.

There are two basic web enrolment pathways - the enrolment of an individual client, and the enrolment of an employer encompassing a group of employees.

Firstly, however, consideration needs to be given to authentication and tokens in the web enrolment process.

Authentication and Tokens

In a web enrolment application, the end user may be someone who is hitherto unknown, so generally, they are not asked to authenticate with the API. Instead, the web application authenticates using staff or admin credentials.

The token obtained in this manner is used as normal (see page 7) in calls to web enrolment functions.

Individual Client Web Enrolment

This scenario involves an individual, (a ‘web client’), enrolling in an occurrence.

Functions like `GetWebProgrammeTypes, GetWebProgrammesForDate` and `GetWebOccurrencesForProgrammeAndDate` are available to offer the user details of which occurrences are available for enrolment.

The user submits a form to the web application, which contains all of the data necessary to make the enrolment. After the web app validates the user input, the following functions are called:

`AddClientWebEnrolment`

Creates skeleton records for the Web Client and the Web Enrolment.

`UpdateWebEnrolment`

Adds details of the Web Enrolment.

`UpdateWebClient`

Adds details of the Web Client.

`UpdateWebClientAVETMISS`

Adds details of the Web Client's AVETMISS information.

`AddWebPayment`

Adds details of any payment for the Web Enrolment. This function would be called after a payment was processed by the web app via a third party payment provider (eg PayPal).

If some condition necessitates cancellation of the web enrolment (and the related web client), a function `DeleteWebEnrolment` is available.

Employer Web Enrolment

This scenario involves an employer or company (a 'web employer') enrolling in an occurrence.

After an employer makes a web enrolment, multiple individuals (web employees) can be added to this single employer web enrolment.

After the interaction between the user and the web app, calls to the API would be something like:

`AddEmployerWebEnrolment`

Creates skeleton records for the Web Client and the Web Employer.

`UpdateWebEnrolment`

Adds details of the Web Enrolment.

`UpdateWebEmployer`

Adds details of the Web Employer.

`AddWebPayment`

Adds details of any payment to the Web Enrolment. This function would be called after a payment was processed by the web app via a third party payment provider (eg PayPal).

Then, for each employee participating in the web enrolment:

`AddWebEmployee`

Adds details of a Web Employee.

`UpdateWebEmployee`

Adds further details of the Web Employee.

`UpdateWebEmployeeAVETMISS`

Adds details of the Web Employee’s AVETMISS information.

A function `RemoveWebEnrolment` is available should the web enrolment and all of its components need to be removed.

Processing User Input

It is anticipated that an end user’s interaction with the web application would be completed before any calls to the API were made, so that the web app would have all required data (and perform any desired data validation) before calling any API functions.

Be aware that there are no facilities to change data in a web enrolment. If any changes or subsequent payments were required, they would need to wait until the web enrolment was processed into VETtrak. The updates would then be performed directly in VETtrak.

The Enrol API does not make any direct changes to the VETtrak data. All changes are processed through the holding table.

There may be some merit in having the web app advise a staff member of new web enrolments, so the web enrolments can be integrated into VETtrak in a timely manner.

Another point to note is that all parties involved in a web enrolment are treated as new. It is not possible to use an existing web client, web company or web employee in a new web enrolment. In other words, a separate web client is created for each client web enrolment, and a separate web employer (with separate web employees) is created for each employer web enrolment. Staff members can manually match these up to existing clients/employers in VETtrak as part of the validation process.

Keep in mind that the above refers to interactions between the web app and the API. How the web app interacts with the end user is entirely in the hands of the web developer.

Accessing API Documentation

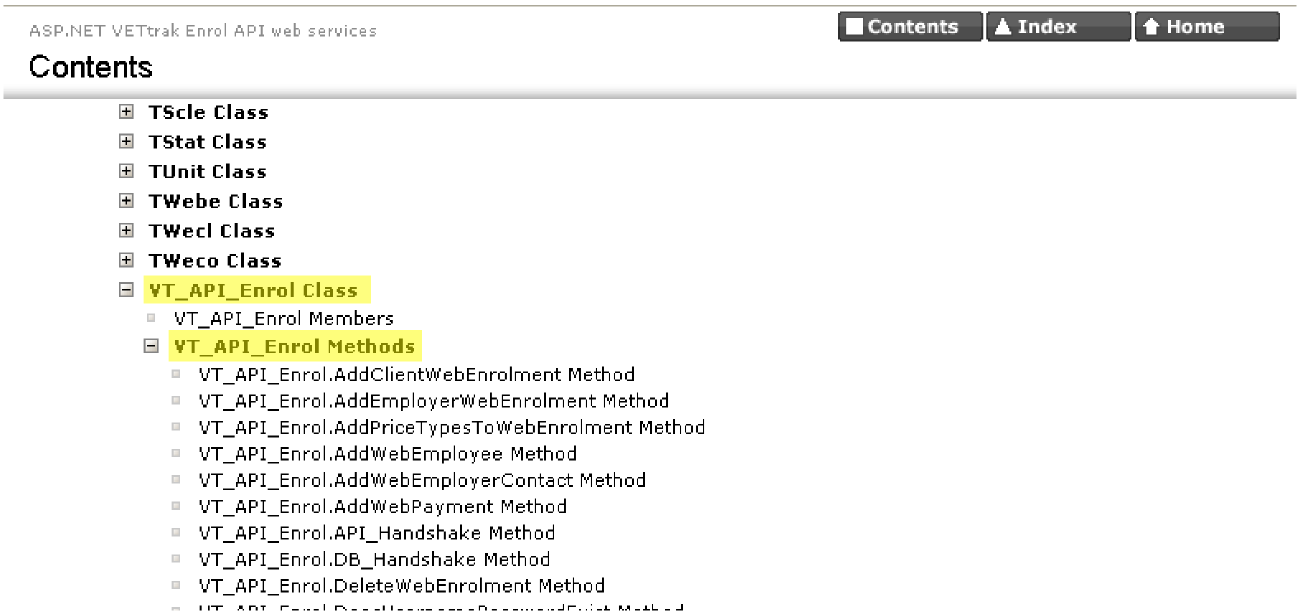
A set of html API documents is available separately.

These have been posted on the VETtrak website at *http://www.vettrak.com.au/api/*.

Functions Documentation

To access documentation on the various functions:

- Click on the link for the appropriate API (Enrol, Portal or Integrate).
- Click on the ***Contents*** button.
- Expand ***Classes***.
- Expand VT_API_XXXX Class (where XXXX is the name of the particular API)
- Click VT_API_XXXX Methods
- Select a function to view full documentation.



Contents

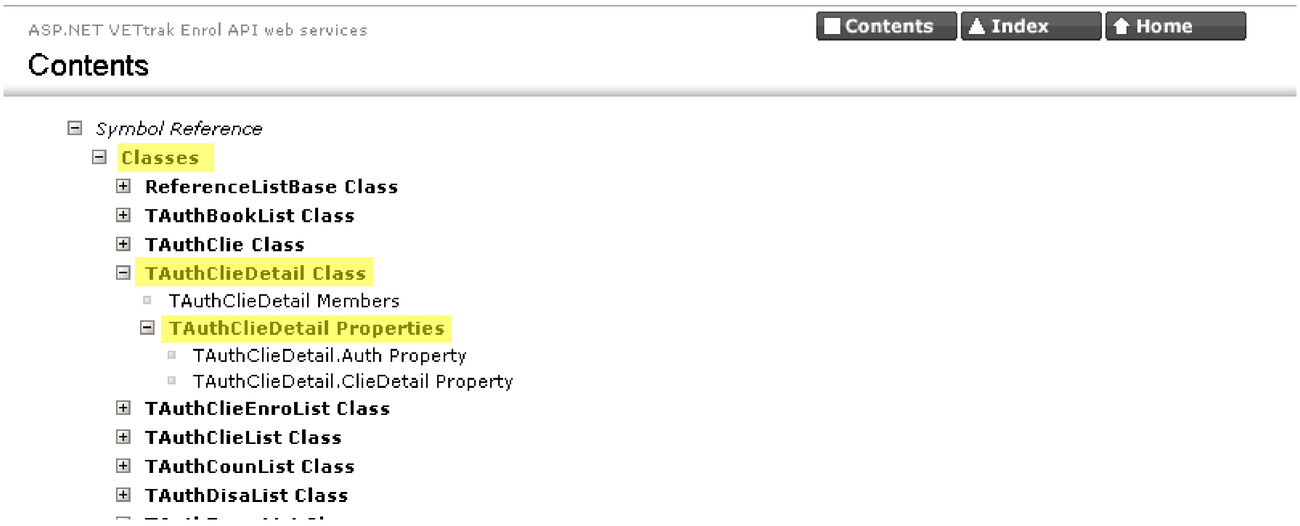
- TScle Class
- TStat Class
- TUnit Class
- TWebe Class
- TWecl Class
- TWeco Class
- VT_API_Enrol Class
  - VT_API_Enrol Members
  - VT_API_Enrol Methods
    - VT_API_Enrol.AddClientWebEnrolment Method
    - VT_API_Enrol.AddEmployerWebEnrolment Method
    - VT_API_Enrol.AddPriceTypesToWebEnrolment Method
    - VT_API_Enrol.AddWebEmployee Method
    - VT_API_Enrol.AddWebEmployerContact Method
    - VT_API_Enrol.AddWebPayment Method
    - VT_API_Enrol.API_Handshake Method
    - VT_API_Enrol.DB_Handshake Method
    - VT_API_Enrol.DeleteWebEnrolment Method

Classes Documentation

Class definitions and documentation are scattered around the other namespaces.

To access documentation on the various Classes:

- Click on the Contents button.
- Expand a Namespace node.
    - Click Classes to access a list of classes and a brief description of each.
    - Expand the Classes node, and expand one of the classnodes.
    - Click Properties to access a list of properties.
    - Click on a property to see a brief description of it.

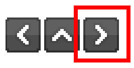

ASP.NET VETtrak Enrol API web services

Contents

- Symbol Reference
    - Classes
        - ReferenceListBase Class
        - TAuthBookList Class
        - TAuthClie Class
        - TAuthClieDetail Class
            - TAuthClieDetail Members
            - TAuthClieDetail Properties
                - TAuthClieDetail.Auth Property
                - TAuthClieDetail.ClieDetail Property
        - TAuthClieEnroList Class
        - TAuthClieList Class
        - TAuthCounList Class
        - TAuthDisaList Class

The index may be helpful if locating the namespace of a particular class proves troublesome. Note that all Classes are preceded with a 'T', which means they are all listed under 'T' in the index.

When viewing a class you can click the 'next' button (located in the top-most frame) in order to view all methods and properties of that class.



Further Information

Links

National training system glossary:

http://www.dest.gov.au/sectors/training_skills/policy_issues_reviews/key_issues/nts/glo/

VETtrak API Documentation:

http://www.vettrak.com.au/api/

VETtrak Product Information:

http://www.vettrak.com.au/products-and-services/vettrak/products-and-pricing

Contact

E: info@vettrak.com.au

P: 03 6333 0166