HP TPware .NET
For HP ACMS and HP TPware

User's Guide

June 2006

The HP TPware .NET User's Guide provides an overview on HP TPware .NET, its installation and migration procedures, and details of the API Client Services that enable connecting a .NET client application to manage ACMS tasks remotely.

Operating System:	Windows 2000, 2003 and XP
Software Version:	HP TPware .NET V5.0
Related Software:	HP TPware Desktop Connector Version 5.0 HP TPware Web Connector Version 5.0 HP ACMS Version 5.0

Copyright Information

Printed in the US

Contents

Tables

Preface

The TPware .NET user's guide provides information on TPware .NET, which is a virtual interface between .NET client applications and ACMS system. Reading the guide enables you to accomplish the following:

- Install TPware .NET for the .NET framework
- Use TPware .NET API Client Services for sending and receiving data between .NET client applications and ACMS system (the communication is established through the TPware gateway server)
- Know the TPware .NET Integration Library and the TPware .NET hierarchy (NameSpace, Classes, Methods and Structures)
- Understand the calling sequence of the TPware API Client Services while executing a task
- Write client applications that use the TPware .NET API Client Services for gaining access and processing information on the ACMS system
- Migrate from an existing TPware environment to the TPware .NET environment

Intended Audience

- Application programmers who need information on TPware .NET API calls while developing client applications using VC++ .NET, VB .NET, C# .NET and J# .NET. The information needed will be on the calling of an appropriate API call for executing a client specific task on the ACMS system.
- Engineers who are responsible for configuring and managing client interfaces developed using TPware .NET.
- Existing TPware engineers who have the .NET framework and are planning to migrate from TPware environment to the TPware .NET environment.

Document Structure

The following table provides the TPware .NET chapter titles and a brief description on each of them:

Table 1.1: Document Structure

Chapter Title	Brief…
Overview	Provides the overview of TPware .NET and the advantages of using the .NET framework and TPware .NET combination.
Installing TPware .NET	Provides the pre-requisites needed and the procedure for installing TPware .NET
TPware .NET API Client Services	Provides information on the following: • TPware .NET Integration Library • TPware .NET hierarchy; includes summary and information on NameSpace, Classes, and Structures; the types of Classes and the various Methods in each Class • Format for every TPware .NET class, method and structure
Programmer's Reference Section	Provides the calling sequence for the TPware .NET Client Services in the Blocking and the Forced Non-blocking mode. Also, a sample client application using the API client services is given.
Migrating from TPware to TPware .NET	Provides the procedure for migrating from TPware to TPware .NET
TPwareReturnValues	Provides the list of TPware return values for every call executed by the client application.

Related Documents

You can refer to the following documents while using TPware .NET:

- *HP TPware Desktop Connector* documentation
- *HP TPware Web Connector* documentation
- *HP ACMS* documentation

Conventions Used

The following table provides the list of conventions used across the document:

Table 1.2: Conventions Used

Convention	Description
italic type	Italic type emphasizes important information, topic and document titles, and error/informational/poll messages.
bold type	Bold type emphasizes new terms, button and file names, and screen names
	The following terms mean the same and are used interchangeably in this document: ● TPware .NET API Client Services and API Client Services ● TPware .NET Integration Library and Integration library
	TPware .NET, ACMS, TPware Desktop Connector and TPware Web Connector are referred to as HP TPware .NET, HP ACMS, HP TPware Desktop Connector and HP TPware Web Connector only in the title page and the first occurrence in the guide.
	The following terms provide different meanings and cannot be interchanged: ● The term SignIn indicates that the call 'SignIn' is being referred to. For example: The .NET TP Desktop/Web Connector client application will call the **SignIn** service to request the TP Desktop Connector Gateway to sign in a user (currently running a desktop client program) into an ACMS system. ● The term sign-in refers to the signing in task where the user logs into OpenVMS or ACMS system. For example: An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in. ● Other such terms are CallId and call-id

Overview

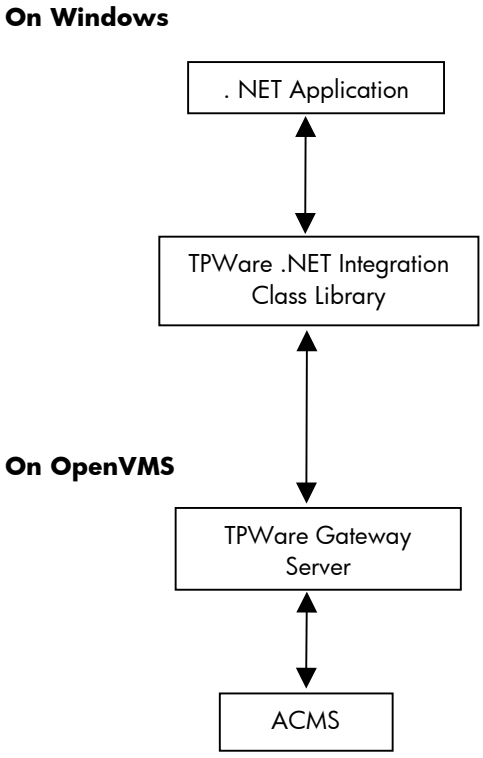
TPware provides an interface between clients running on a PC and the business application. TPware is a set of components from which several TP-related products are created. Different components are used to create different products. These products are as follows:

- HP TP Desktop Connector: The HP TP Desktop Connector product allows you to connect a client application to a TP server application. TPware accomplishes client access to TP applications by enabling the creation of code called a TP stub that translates calls between client programs and TP applications. Using a client program, the user enters a request by entering information in a programmer-designed window. The client program processes the incoming request and extracts information needed to call the TP application.

- HP TP Web Connector: The HP TP Web Connector produces objects that you can easily use to connect a web server to your application. Using TPware technology, HP TP Web Connector software expands the capabilities of World Wide Web servers by enabling them to access ACMS tasks. HP TP Web Connector software accomplishes this connection by enabling the creation of code called a TP stub that translates calls between web servers and TP applications. Using a web browser, the user accesses a page on a web server.

TPware .NET is an interface between the Windows based .NET client applications and the TPware Gateway server, which in turn communicates with the ACMS system. It belongs to the TPware product family and, manages and monitors the ACMS applications from the Windows platform. It has a set of API Client Services that are referenced by the Windows based applications to communicate with the ACMS system on OpenVMS. These API Client Services form the TPware .NET Integration Class Library. The TPware .NET and the TPware Gateway server are the point of contact on the respective Windows and OpenVMS platforms.

TPware .NET Framework



Advantages of using the .NET Framework

- The .NET framework provides rapid application migration and developing new application capabilities on windows.
- .NET support for the ACMS TPware API enables language independence for the API. The API can be used from any .NET-supported language. The .NET framework supports 25 languages for developing applications, which includes:
  - VC++ .NET
  - VB .NET
  - C# .NET
  - J# .NET
- It is also possible to develop web-based applications using ASP .NET with Microsoft IIS server using languages like VB.NET, C# .NET and J# .NET

Supported Features

Following are the features supported in TPware .NET V5.0:

- TPware .NET Integration Library components support the Blocking and Forced Non-blocking communication modes.
- TPware .NET supports the NO I/O mode. In the NO I/O mode, forms are not used and data is passed directly to the ACMS system.

Restrictions

Following are the features not supported in TPware .NET V5.0:

- Regular Non-blocking
- Exchange steps
- Form I/O
- Request I/O
- Portable API Presentation Services

Installing TPware .NET

Introduction

TPware .NET plug-in is a freeware and can be downloaded from the TPware website. Refer to the following location to access the download. Ensure that the following pre-requisites are installed before downloading the freeware:

Note: If you are already using TPware Desktop Connector or Web Connector and want to use TPware .NET, then refer to Migrating to TPware .NET for details on the migration procedure.

Pre-requisites

- **Software**
  - TPware kit
    - For more information, refer to the following documents:
      - *HP TP Desktop Connector for ACMS Installation Guide*
      - *HP TP Web Connector for ACMS Installation Guide*
  - .NET Framework, version 1.1 or later
  - Windows 2000, Windows XP or Windows 2003

For more information, refer to the following documents:

- HP TP Web Connector V5.0 SPD 64.97.10
- HP TP Desktop Connector V5.0 SPD 34.81.23

Installation Procedure

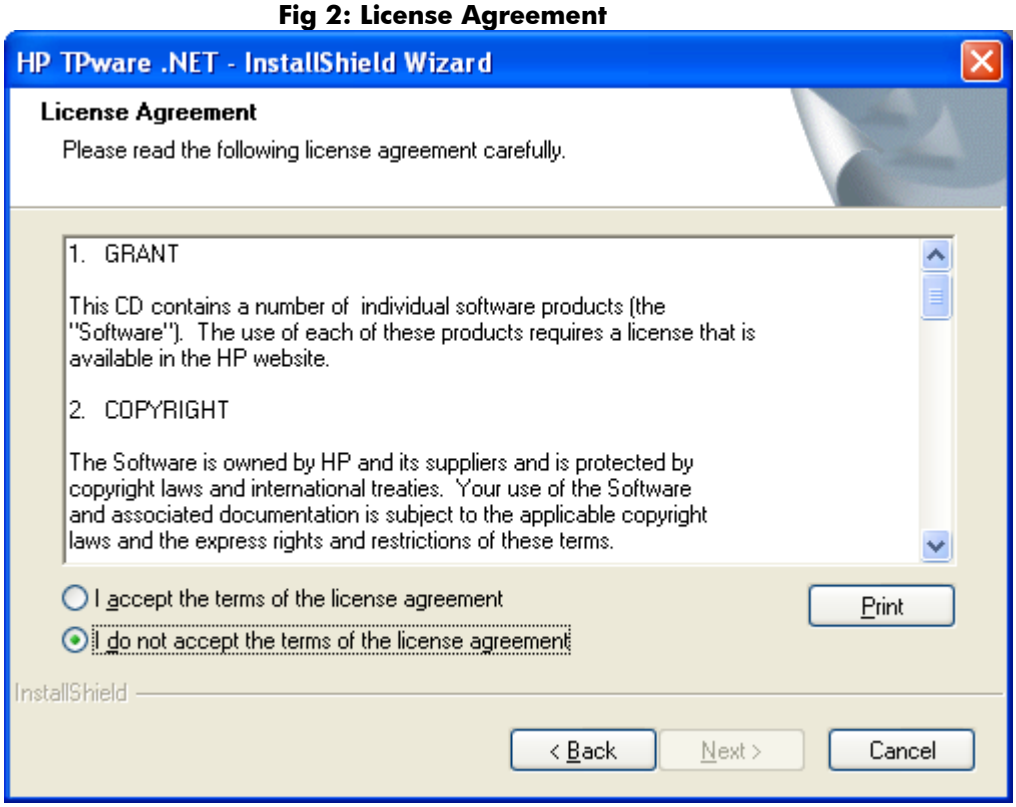
Follow these steps for installing TPware .NET:

1. Run Setup.exe. The following HP TPware .NET InstallShield Wizard screen is displayed. See Fig1: HP TPware .NET InstallShield Wizard

Fig 1: HP TPware .NET InstallShield Wizard

2. Click Next. The License Agreement screen is displayed.

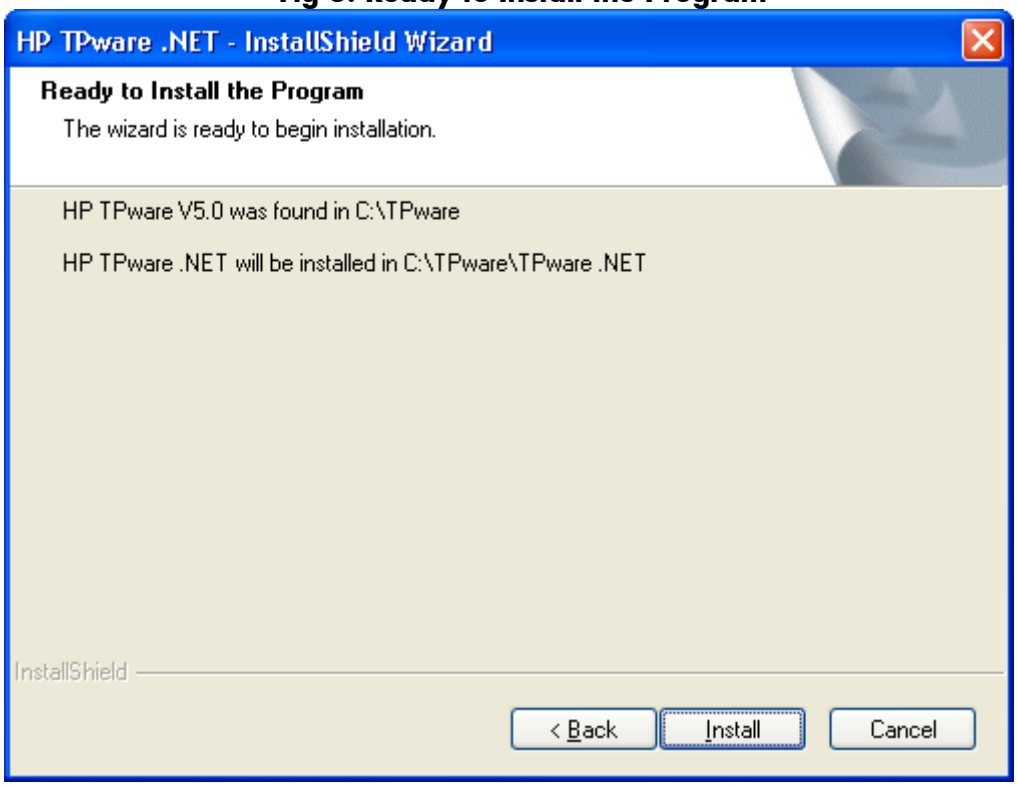
Fig 2: License Agreement



3. Read the license agreement. After accepting the terms and conditions, click Yes to proceed with the installation.
4. The Installer verifies the pre-requisites list. If the pre-requisites are already installed, the next screen is displayed where you have to enter your name and your company name.

5. Click Next. The Installer displays the Ready to Install the Program screen with the location of the latest HP TPware kit. It also informs that HP TPware .NET will be installed in the folder TPware .NET in the same location.

Fig 3: Ready to Install the Program



6. Click Install. The Installer installs TPware .NET in the folder <TPware Kit>\TPware .NET.
   Note: If multiple versions of the TPware kit are installed, then TPware .NET will be installed in the directory structure of the highest installed version.

   The InstallShield Complete Wizard screen is displayed with the information that the installation has been successful. Click Finish to exit the wizard.

TPware .NET API Client Services

Introduction to TPware .NET Integration Library

The TPware .NET Integration Class Library is a collection of classes, methods and structures under a common NameSpace. They are also called the TPware Integration Library components, which can be referenced by any desktop or web based applications using the .NET framework. Each class has specific methods that can be called from any .NET-supported language. The integration library manages the conversion of data to the recipient operating system's data format when transferring data from one source to another.

For example, when the data is transferred from a client application on Windows (source) to an ACMS system on OpenVMS (recipient), the data is converted to the OpenVMS specific data format and vice versa.

Summary of TPware .NET API Client Services

TPware consists of a set of client services that are used to manage ACMS applications. These client services on Windows communicate with the TPware Gateway on OpenVMS, which in turn communicates with ACMS.

Communication Types

The following communicating channels are used to execute TPware Client Services' calls:

- Blocking mode: In the Blocking mode, a particular call is completed before the next call is executed. All data to be returned for each call is returned immediately on completion of the call.
- Non-blocking mode: In the Non-blocking mode, the call returns with a call submission status information and does not wait for completion. The client application has to poll periodically for the status of the submitted activity. The return data for a specific call is returned after the call is completed. In the non-blocking mode, all workspace data is passed in the task invocation call. These include any workspaces that are to be returned. Memory should be allocated in the client application for the return data also and this memory should not be freed until the specified call completes.

  Following are the two types of Non-blocking modes:

  - Regular Non-blocking mode: A completion routine is provided in this mode, which is invoked by the TPware client as soon as the submitted activity is completed. This completion routine is triggered by the completion status when the polling call is made.
  - Forced Non-blocking mode: In this mode, no completion routine is provided. Applications using forced non-blocking calls have to poll periodically to check the status of the submitted activity in this mode also. If the status is completed, then a completion call has to be to complete the specified call and acknowledge the status.

Supported Client Services

Following are the two supported client services:

- Portable API Client Services: The TP Desktop Connector Portable API Client Services allow you to write a desktop client program on desktop systems without extensive knowledge of network communications. The table below summarizes the portable client API services and their support in TPware .NET.

- Forced Non-blocking Client Services: The Forced Non-blocking client services extend the Portable API to support development tools that do not provide for callbacks, data pointers or consistent memory locations for data. (Such tools include Visual Basic and others.) You create a forced non-blocking session when you specify the TPware.Option, TPware.Opt.NonBlk (ACMSDI_OPT_NONBLK), with the TPware.ClientServices.SignIn service and do not supply a completion address. The Forced non-blocking client services and their support in TPware .NET are summarized below.

Hierarchy of TPware Client Services' Calls

Following is the hierarchy for the TPware .NET API Client Services. Click each link to view the class/method/structure introduction, format, parameters given and the values returned:

TPware

- TPware.ClientServices
  - TPware.ClientServices.SignIn
  - TPware.ClientServices.SignOut
  - TPware.ClientServices.CallTask
  - TPware.ClientServices.Cancel

- TPware.ForcedNonBlockingClientServices
  - TPware. ForcedNonBlockingClientServices.CompleteCall
  - TPware.ForcedNonBlockingClientServices.Poll

- TPware.LoginCredentials
- TPware.SubmitterId
- TPware.CallId
- TPware.Workspace
- TPware.OptionType
- TPware.Option
- TPware.PwdExpiry
- TPware.ApplicationDetails
- TPware.TPwareReturnValue

Table 1.3: TPware .NET API Client Services

Term	Description
NameSpace	The NameSpace for all the TPware classes and structures in the Integration Class Library is **TPware**.
Classes	Following are the classes under the TPware NameSpace: **Class Name** — **Defined as...** ClientServices — TPware.ClientServices ForcedNonBlockingClientServices — TPware.ForcedNonBlockingClientServices
Methods	Following are the methods under the **ClientServices** class: **Method Name** — **Defined as...** SignIn — TPware.ClientServices.SignIn SignOut — TPware.ClientServices.SignOut CallTask — TPware.ClientServices.CallTask Cancel — TPware.ClientServices.Cancel Following are the methods under the **ForcedNonBlockingClientServices** class:

Term	Description	
	Method Name	**Defined as...**
	CompleteCall	TPware.ForcedNonBlockingClientServices.CompleteCall
	Poll	TPware.ForcedNonBlockingClientServices.Poll
Structures	Following are the structures under the TPware NameSpace:	
	Structure Name	**Defined as...**
	SubmitterId	TPware.SubmitterId
	CallId	TPware.CallId
	Workspace	TPware.Workspace
	OptionType	TPware.OptionType
	Option	TPware.Option
	PwdExpiry	TPware.PwdExpiry
	ApplicationDetails	TPware.ApplicationDetails
	TpwareReturnValue	TPware.TPwareReturnValue

The following blocks describe each class and its succeeding methods in detail followed by a separate block on TPware structures.

Client Services

Introduction

Client Services class will provide methods for all the API client services of TPware exposed by the TPware .NET integration code. Following will be the different methods:

SignIn

The .NET TP Desktop/Web Connector client application will call the SignIn service to request the TP Desktop Connector Gateway for signing in a user (currently running a desktop client program) into an ACMS system.

Format

```
Public TPwareReturnValue SignIn (
TPware.LoginCredentials   Credentials,
out TPware.SubmitterId    Submitter,
TPware.Option[]           Options,
ref int                   CompStatus
)
```

Parameters

Credentials	An object of type TPware.LoginCredentials that will have the login credentials required for authentication.
Submitter	An object of type TPware.SubmitterId that will have the submitter ID. It will be written into by the sign-in call and passed by reference as an output parameter.
Options	An Object of type TPware.Option that will have a list of options for the session.
CompStatus	An integer where the completion status of the SignIn request will be written. It will be passed by reference.

Return Value

It will contain an object of type TPwareReturnValue; it will return the status value having the result of the SignIn operation.

SignOut

The .NET TP Desktop/Web Connector client application will call the SignOut service to request the TP Desktop Connector gateway for terminating an active session with the ACMS system. It will ensure that all network links are properly shut-down.

Format

```
Public TPwareReturnValue SignOut (
TPware.SubmitterId Submitter,
ref int CompStatus
)
```

Parameters

Submitter	An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in.
CompStatus	An integer where the completion status of the SignOut request will be written. It will be passed by reference.

Return Value

It will contain an object of type TPwareReturnValue; it will return the status value having the result of the SignOut operation.

CallTask

The .NET TP Desktop/Web Connector client application will call the CallTask service to execute a task in an ACMS application. The client application will send a request to the TP Desktop Connector gateway to start the task either in a blocking or non-blocking mode. During the task execution, the TP Desktop Connector gateway will manage the Exchange step processing by calling the customer-written generic presentation procedures in the desktop client program.

Format

```
Public TPwareReturnValue CallTask (
TPware.SubmitterId          Submitter,
TPware.ApplicationDetails   AppDetails
ref Stack                   WkspStack,
out TPware.CallId           CurrCallId,
ref int                     CompStatus,
out IntPtr                  WkspData
)
```

Parameters

Submitter	An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in.
AppDetails	An object that will have the details of the application and the task that needs to be executed.
WkspStack	A stack that will have the workspace structures to be passed to an ACMS task. The objects will be passed by reference in the sequence in

	which they are put on the stack. It implies that the first object put on the stack will be passed first and the last object will be passed last.
CurrCallId	An object of type TPware.CallId where the CallTask service will write newly created unique call identification. It will be used by the Submitter to identify an active call and will be passed as an output parameter.
CompStatus	An integer where the completion status of the CallTask request will be written.
WkspData	A pointer to the workspace data in an unmanaged space. It will point to the updated workspace data after the task completion and will be passed by reference as an output parameter.

Return Value

It will contain an object of type TpwareReturnValue; it will return the status value having the result of the CallTask operation.

Cancel

The .NET TP Desktop/Web Connector client application will call the Cancel service to cancel a currently active ACMS task. It will be used only in the Non-blocking mode to cancel a task executed using a non-blocking service.

Note: Do not use the Cancel service from an asynchronous completion routine.

Format

```
Public TPwareReturnValue Cancel (
TPware.SubmitterId      Submitter,
TPware.CallId           CurrCallId,
int                     Reason,
ref int                 CompStatus
)
```

Parameters

Submitter	An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in.
CurrCallId	An object of type TPware.CallId that will have the call-id for the task that is being canceled. It will be the same call-id that would have been passed back on the CallTask service as an output parameter.
Reason	An integer value that will specify the status value of the reason code for the Cancel request. The TP Desktop Connector gateway will pass the value to the Application Execution Controller (EXC). The default will be - ACMSDI_CALL_CANCELED, "the task was canceled by the task submitter".
CompStatus	An integer where the completion status of the CallTask request will be written. It will be passed by reference.

Return Value

It will contain an object of type TpwareReturnValue; it will return the status value having the result of the Cancel operation.

Forced Non-blocking Client Services

Introduction

Forced Non-blocking Client Services class will provide methods for all the forced-Non-blocking client services of TPware exposed by the TPware .NET integration code. Following will be the different methods:

CompleteCall

The .NET TP Desktop/Web Connector client application will require the CompleteCall service to obtain the completion arguments for CallTask, SignIn, SignOut, and Cancel services. When Poll (another Forced Non-blocking Client Service) will detect task completion, the completion status for these services will be passed to CompleteCall. For the CallTask service, the CompleteCall could also obtain the TPware ACMS status message and task argument workspaces sent from TPware gateway.

Format

```
Public TPwareReturnValue CompleteCall (
TPware.SubmitterId          Submitter,
TPware.CallId               CurrCallId,
ref int                     CompStatus,
bool                        TaskCall,
ref Stack                   Wksp,
IntPtr                      WkspData
);
```

Parameters

Submitter	An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in.
CurrCallId	An object of type TPware.CallId that will have the call-id for the task that is being canceled. It will be the same call-id that would have been passed back on the CallTask service as an output parameter.
CompStatus	An integer where the completion status of the CompleteCall request will be written. It will be passed by reference.
TaskCall	A Boolean used to identify whether the CompleteCall service will be called after the CallTask service or the SignIn/SignOut/Cancel services. This flag will be used to determine whether the workspace data should be or should not be retrieved during the call completion.
Wksp	A stack that will have the workspace structures to be passed to an ACMS task. The objects will be passed in the sequence in which they will be put on the stack. It implies that the first object put on the stack will be passed first and the last object will be passed last. They will be passed by reference.
WkspData	A pointer to the workspace data in an unmanaged space. It will point to the updated workspace data after the task completes. It will be passed as an output parameter.

Return Value

It will contain an object of type TPwareReturnValue; it will return the status value having the result of the CompleteCall operation.

Poll

The .NET TP Desktop/Web Connector client application will call the Poll service to check for and process messages sent from a TP Desktop/Web Connector gateway. These messages will be sent to an active submitter in the desktop application. The application will have to periodically call this service in a forced non-blocking environment to check for completion of outstanding SignIn, CallTask, SignOut, and Cancel requests.

Format

```
Public TPwareReturnValue Poll (
TPware.SubmitterId        Submitter,
)
```

Parameters

Submitter	An object of type TPware.SubmitterId that will have the Submitter ID used for the sign-in.

Return Value

It will contain an object of type TPwareReturnValue; it will return the status value having the result of the Poll task.

TPware .NET Data Structures

Introduction

TPware .NET data structures, a collection of objects will be used to send data that can be specifically interpreted by the TPware 'C' language. They could be directly associated with the TPware NameSpace.

LoginCredentials

The LoginCredentials will contain the required credentials for a TPware client login. The following table will list the LoginCredentials attributes:

Table 1.4: LoginCredentials Attributes

Field Name	Type	Description	Access
Server	String	It will contain the IP Address or the host name of the TPware gateway server.	Public
UserName	String	It will contain the user name of the user on the TPware gateway server. The user should also be an ACMS user.	Public
Password	String	It will contain the password of the user on the TPware gateway server. The user should also be an ACMS user.	Public

SubmitterId

The SubmitterId will contain the Submitter (or Submitter's) identification. The following table will list the SubmitterId attributes:

Table 1.5: SubmitterId Attributes

Field Name	Type	Description	Access
Submitter	IntPtr	It will be a pointer pointing to the SubmitterId.	Public
Instance	short	It will be the instance number for this (active) submission.	Public

CallId

The CallId will contain the identification information for a specific call submitted using the CallTask. The following table will list the Call Id attributes:

Table 1.6: Call Id Attributes

Field Name	Type	Description	Access
Call	IntPtr	It will be a pointer that points to the Call Id.	Public
Instance	short	It will be the instance number for this call.	Public
OwnerIsAcmsdi	short	It will specify whether the owner of this call is an acmsdi or not.	Public

Workspace

The Workspace will contain information about the workspace that will be passed to an ACMS application. The following table will list the Workspace attributes:

Table 1.7: Workspace Attributes

Field Name	Type	Description	Access
Length	Int32	It will contain the workspace length.	Public
Data	IntPtr	It will contain a pointer to the workspace data.	Public

PwdExpiry

The PwdExpiry will contain details for the password expiry option. The following table will list the PwdExpiry attributes:

Table 1.8: PwdExpiry Attributes

Field Name	Type	Description	Access
Option	OptionType	It will point to the specified option, ACMSDI_OPT_PWD_EXPIRING.	Public
Address	IntPtr	It will point to a long integer containing the frequency in hours for which password expiry validation must be completed.	Public

Option

The Option will behave like a union and contain the list of TPware options for a session. Since C# does not support unions, field offsets will be used to derive the behavior of a union. The following table will list the Option attributes:

Table 1.9: Option Attributes

Field Name	Type	Description	Access	Field Offset
Option	OptionType	It will point to the specified option.	Public	0
PwdExpiringHrs	PwdExpiry	It will point to the password expiry structure.	Public	0

ApplicationDetails

The ApplicationDetails will contain details of the application as well as the task that needs to be executed. The following table will list the ApplicationDetails attributes:

Table 1.10: ApplicationDetails Attributes

Field Name	Type	Description	Access
ApplicationName	String	It will be the name of the ACMS application in which the task to be executed resides.	Public
TaskName	String	It will be the name of the task to be executed.	Public
SelectionString	String	It will be used to pass any additional information to the task.	Public

OptionType

The OptionType will contain the various option types that could be passed for a TPware session. The following table will list the OptionType attributes:

Table 1.11: OptionType Attributes

Field Name	Value	Description	Access
OptEndList	0	It will be the delimiter for the list and points to the end of the list.	Public
OptPwdExpiring	4	It will enable checking for passwords that are about to expire.	Public
OptNonBlk	5	It will enable the forced non-blocking mode.	Public

TPwareReturnValue

The TPwareReturnValue will provide the return value for every call executed by the desktop client application. Every TPwareReturnValue will be given a unique field name and appropriate description.

Table 1.12: TPwareReturnValues

Field Name	Value	Description
Normal	0	The *Normal Completion* message will be displayed if the Desktop ACMS service completes without giving any error.
ApplDead	-3001	The *ACMS Application Not Started* error will be displayed when the ACMS Desktop Adapter Gateway calls ACMS to execute an ACMS application and the executed application stops unexpectedly.
CallActv	-3002	The *Call Active – Cannot Start New Operation* error will be displayed if a client application requests the TPware gateway for executing a task or signing out of an ACMS task and during the process of executing one of these requests, the TPware gateway detects that a task is currently active for that specific Submitter ID. **Note**: A single submitter ID can have only one active task at a time and cannot sign-out when that task is active.
InsufPrm	-3003	The *Insufficient Parameters* error will be displayed if the Desktop ACMS service was called with insufficient parameters or not allocating sufficient memory. It could occur as a result of programming errors, such as the client program not allocating memory for Desktop ACMS structures passed as parameters. It could also occur as a result of truncation in development tools such as Visual Basic.
Internal	-3004	The *Internal* error will be displayed while executing a Desktop ACMS service and an error was encountered in the client services, or the Desktop ACMS gateway.
InvCallId	-3005	The *Invalid Call Identification* error will be displayed if the client program had called TPware.ForcedNonBlockingClientServices.CompleteCall_PP using an invalid call-id.
InvLogin	-3006	The *Invalid Login Attempt* error will be displayed if the gateway received a sign-in request from a desktop client and was unable to sign the user into OpenVMS or ACMS system.
InvOption	-3007	The *Invalid Option in Desktop ACMS Service* error will be displayed if a Desktop ACMS service is called with an invalid value in the sign-in option or the call option parameter.
InvSubId	-3008	The *Invalid Submitter Identification* error will be displayed if a Desktop ACMS service is called with an invalid Desktop ACMS Submitter ID.
MixedMode	-3009	The *Using Both Blocking and Non-blocking Modes* error will be displayed in the following scenarios: • A Desktop ACMS service will be called in the Blocking mode when a previous call to the same gateway by the same client program will be called using the Non-blocking mode. • A Desktop ACMS service will be called in the Non-blocking mode when a previous call to the same gateway by the same client program will be called using the Non-blocking mode.

Field Name	Value	Description
NoAcms	-3010	The *ACMS Not Active* error will be displayed if the gateway attempts to sign-in a user to ACMS when the ACMS system is not active.
NoMemory	-3011	The *Low Memory Resource* error will be displayed if a Desktop ACMS service is unable to execute because of not being able to allocate sufficient memory on the client system.
NoSuchAppl	-3013	The *ACMS Application Not Found* error will be displayed if the Gateway calls ACMS to start a task and ACMS in turn returns the ACMS$_APPL__NOT_STARTED message. The message indicates that either there is an error in the application name or that the application name is valid but it is not currently active on the requested node.
NoSuchTask	-3014	The *Invalid Task Code* error will be displayed if the Gateway calls ACMS to start a task and ACMS in turn returns "Could not find a task of that name in the specified application". The message indicates that either there is an error in the task name or that an irrelevant application is specified for the task.
OprCancelled	-3015	The *Operator Cancelled ACMS User* error is displayed if during the execution of a task or sign-out request, the ACMS/CANCEL USER command is used to cancel the Submitter. If a task is active at that point in time, then that task is cancelled and the Submitter is also signed out.
Pending	-3016	The *Operation Started Status* message will be displayed only by services that are executed in a Non-blocking mode. The 'Pending' status indicates that the service called by the client program is active.
SecChk	-3017	The *ACMS Task ACL Failure* error will be displayed if the client program requests a task that the user is not authorized to execute: Following are the scenarios: • If the application having the task is on the same OpenVMS system as that of the Desktop ACMS Gateway, the EXC (Execution) performs the access control list check using the OpenVMS username of the user currently signed in. • If the application is on a different OpenVMS system from that of the Desktop ACMS Gateway, the EXC performs the access control check using either of the following: – Assigned proxy for the username on the Desktop ACMS Gateway code – Default proxy defined in ACMSGEN – Username (if neither proxy exists)
SignInActv	-3018	The *Sign-in in Progress* status message will be displayed if the client program requests a task or a sign-out while the sign-in for that Submitter ID is still in progress.
SignOutActv	-3019	The *Sign-out in Progress* status message will be displayed if the client program requests a sign-out while the sign-out for that Submitter ID is already in progress.
SrvDead	-3020	The *Desktop ACMS Gateway has Died* error will be displayed for sign-in requests in the following scenarios: • If the network link cannot be established or the Desktop ACMS gateway is not available on the node specified in the sign-in request. For other Desktop

Field Name	Value	Description
		• If the network link has failed or the Desktop ACMS gateway is no longer available on the node to which the user has currently signed in.
TaskAbort	-3021	The *Task Aborted* error will be displayed when a task is executed and it completes abnormally. It can be due to an ACMS error or other non-recoverable conditions.
TaskCancelled	-3022	The *Task Cancelled by Operator* error will be displayed when a task is executed and the ACMS/CANCEL TASK command is used to cancel the task.
TaskSpDied	-3023	The *Task Procedure Gateway Process has Died* error will be displayed when a task is executed and it fails in a processing step because a procedure gateway process has stopped. Typically the stopping of the procedure process is caused by a fatal error in the step procedure that causes a rundown of the procedure gateway process.
TaskFailed	-3024	The *Task Failed to Complete Normally During a Task Call* error will be displayed for varied reasons, some of them are listed: • User-initiated cancels accomplished through CANCEL or EXCEPTION syntax in the task definition • Cancels initiated in a processing step (using ACMS$REQ_CANCEL or RDB$SIGNAL) • EXC-initiated cancels because of an exchange step failure
InvProtocol	-3025	The *Protocol Versions of the Client and the Gateway Softwares do not Match* error will be displayed while processing a sign-in request. During the processing of the request, the Desktop ACMS Gateway detects that the client services software version does not match with the Gateway software version.
BadNodeName	-3026	The *Node/Host Name Lookup Failure* error will be displayed if the network services are not able to locate the node specified on TPware.ClientServices.SignIn.
PwdExpired	-3027	The *Password has Expired* error will be displayed when the Gateway receives a sign-in request from a desktop client and is unable to sign the user into ACMS system because of password expiry for the user's account.
CancelActv	-3028	The *Client-initiated Cancel in Progress* status message is displayed when a Desktop ACMS service is called while a TPware.ClientServices.Cancel call is in progress. The service requested cannot be handled by Desktop ACMS client service during the processing of a client-initiated cancel request.
Unsupported	-3031	The *Function or Option Not Supported on the Selected ACMS Desktop Gateway* error will be displayed when the client application has issued a call to an ACMS Desktop back-end Gateway, which in turn requests execution of a function not supported or contains an option not supported.
CnvFailed	-3032	The *Character Set Conversion Routine has Failed* error will be displayed when a special routine installed to perform conversion between client-based and gateway-based character sets' fails. An example of a character set conversion routine: conversion of Kanji characters to Simple Latin and vice-a-versa.

Field Name	Value	Description
PollActv	-3033	The *TPware.ForcedNonBlockingClientServices.Poll Call in Progress* error will be displayed when a Desktop ACMS service is called simultaneously while a TPware.ForcedNonBlockingClientServices.Poll call is in progress and the requested service cannot be handled by the Desktop ACMS client services.
WrongState	-3043	The *Session is in the Wrong State for this Call* error will be displayed in the Forced Non-blocking environment when the state of the current session is not valid for the call issued.
CmpclActv	-3044	The *Request is Invalid while Complete Call is Active* error will be displayed when the client program requests an operation while a TPware.ForcedNonBlockingClientServices.CompleteCall is still in progress.
BindSesIdActv	-3045	The *ACMSDI_BIND_SESSION_ID Call in Progress* error will be displayed when the client program requests an operation while an acmsdi_bind_session_id call is still in progress.
PwdExpiring	-3100	The *Number of Hours Returned Until Password Expires* error will be displayed when the gateway receives a sign-in request from a desktop client and succeeds in signing the user into OpenVMS and ACMS systems. However, upon completing the OpenVMS sign-in, the Desktop ACMS gateway detects that the password will expire in lesser time than the password expiration threshold value specified when starting the Gateway. The Gateway calculates the number of hours until expiration, and returns the value in the sign-in options array (for TPware.ClientServices.SignIn) or the DBGetErr item2 field (for DBInit).
CallCanceled	-3101	The *Task Submitter Cancelled the Task* error will be displayed when the client application issues a TPware.ClientServices.Cancel call that is successful, causing the ACMS task to be cancelled.
Ready	-3102	The *No Call Executing and No Message Available* status message will be displayed when there are currently no calls executed and there are no messages from the gateway available to be processed. TPware.ForcedNonBlockingClientServices.Poll gives the message.
Exec	-3103	The *Call is Executing; No Message Currently Available* status message will be displayed when a call to the gateway is being executed but there is no message from the gateway available to be processed. TPware.ForcedNonBlockingClientServices.Poll gives the message.
Done	-3104	The *Sign-in, Sign-out or Task-call Completion* status message will be given by the gateway and returned by the TPware.ForcedNonBlockingClientServices.Poll. It shows that the gateway has responded with a message indicating the completion of a TPware.ClientServices.SignIn, a TPware.ClientServices.SignOut or a TPware.ClientServices.CallTask call.
CancelDone	-3105	The *Task Cancel Completion* status message will be given by the gateway and returned by the TPware.ForcedNonBlockingClientServices.Poll. It shows that the gateway has responded with a message indicating the completion of a TPware.ClientServices.Cancel call.

Programmer's Reference Section

Introduction

This section provides a detailed description of the order in which the TPware .NET Client Services are called when a particular task is executed by a client application. The order in which they are called depends on the mode chosen by the client applications for executing a particular task. Following are the two modes and the order in which the TPware .NET Client Services are called:

In Blocking Mode

In Blocking mode, the user waits for the completion of the executed task. All the data that needs to be returned to the calling client application is immediately returned on the completion of the call. Following is the sequence in which the TPware .NET Client Services are called in the Blocking mode:

1. The user sends a request to the TP Desktop Connector gateway for providing access (sign in) to the ACMS system.
   Call name: SignIn
2. The user then sends a request to the TP Desktop Connector gateway to start an application specific task on the ACMS system.
   Call name: CallTask
3. Once the intended task is completed, the user sends a request to the TP Desktop Connector gateway for logging out of the ACMS system.
   Call name: SignOut

In Forced Non-blocking Mode

In Forced Non-blocking mode, the user does not wait for the completion of the executed task but the user has to periodically poll the status of the executed task. Once all the data that needs to be returned to the calling client application is returned, the user has to complete the specified call/task and acknowledge the status of the call. Following is the sequence in which the TPware .NET Client Services are called in the Forced Non-blocking mode:

1. The user sends a request to the TP Desktop Connector gateway for providing access (sign in) to the ACMS system.
   Call name: SignIn
2. The user periodically polls the TP Desktop Connector gateway to know the status of the sign in request.
   Call name: Poll
3. If the status of the polling is positive, the user does the following:
   a. Completes the call that was used to sign into ACMS system
   b. Acknowledges the status of the completed task (sign in request) that is got through polling.
   Call name: CompleteCall (the use of CompleteCall confirms that the user can access the ACMS system.)

   Note: If the status of the polling request is negative, then the user continues to poll the TP Desktop Connector gateway.
4. The user then sends a request to the TP Desktop Connector gateway to start an application specific task on the ACMS system.
   Call name: CallTask
5. The user repeats step 2 to know the status of the call request, step 3 to complete the call and acknowledge the status of the call request.
   Call names: Poll and CompleteCall

6. Once the intended task is completed, the user sends a request to the TP Desktop Connector gateway for logging out of the ACMS system.
   Call name: SignOut
7. The user repeats step 2 to know the status of the call request, step 3 to complete the call and acknowledge the status of the call request.
   Call names: Poll and CompleteCall

Following is the sequence in which the client services are called incase the user wants to cancel the task that is executed:

1. The user sends a request to the TP Desktop Connector gateway for providing access (sign in) to the ACMS system.
   Call name: SignIn
2. The user periodically polls the TP Desktop Connector gateway to know the status of the sign in request.
   Call name: Poll
3. If the status of the polling is positive, the user does the following:
   a. Completes the call that was used to sign into ACMS system
   b. Acknowledges the status of the completed task (sign in request) that is got through polling.
   Call name: CompleteCall (The use of CompleteCall confirms that the user can access the ACMS system.)

   Note: If the status of the polling request is negative, then the user continues to poll the TP Desktop Connector gateway.
4. The user sends a request to the TP Desktop Connector gateway for canceling the task that is executed on the ACMS system.
   Call name: Cancel
5. The user repeats step 2 to know the status of the call request, step 3 to complete the call and acknowledge the status of the call request.
   Call names: Poll and CompleteCall

Sample TPware .NET Code

The TPware .NET samples demonstrate the simplicity of building desktop and web applications that call ACMS tasks using the HP TPware .NET. The following samples are included with HP TPware .NET kit:

- CONFERENCE_BOOKING_APPL sample
- STUDENT_APPL sample

Note: These sample applications' work only on OpenVMS Alpha and I64.

The CONFERENCE_BOOKING_APPL Sample

The CONFERENCE_BOOKING_APPL server sample that runs on your OpenVMS host consists of a conference room booking application. The application consists of 13 ACMS tasks that provide the functionality for the following facilities:

- Adding/deleting conference room
- Adding/deleting employee name
- Viewing existing conference room booking
- Booking conference room
- Cancellation of conference room booking

The CONFERENCE_BOOKING_APPL sample consists of the following components:

- An OpenVMS saveset file containing the files for the ACMS application definitions and source code
- A self extracting exe archive file containing a Visual C# .NET GUI application that demonstrates calling tasks using TPware .NET
- A self extracting exe archive file containing a Visual Basic .NET GUI application that demonstrates calling tasks using TPware .NET

To begin with, you should build the CONFERENCE_BOOKING_APPL server sample. When the server application is completely built, you can proceed to building the CONFERENCE BOOKING APPL desktop applications and then you can execute the applications. Following is the sequence in which the sample applications are built and executed:

1. Building, Installing and Running the CONFERENCE_BOOKING_APPL Server
2. Starting the ACMS Application
3. Building the CONFERENCE_BOOKING_APPL Desktop Application
4. Building the CONFERENCE_BOOKING_APPL C# .NET Application
5. Building the CONFERENCE_BOOKING_APPL Visual Basic .NET Application
6. Running the Applications

Building, Installing and Running the CONFERENCE_BOOKING_APPL Server

The CONFERENCE_BOOKING_APPL server is built on the OpenVMS system.

Note: Ensure that an ACMS development system is already configured and OpenVMS components of the HP TP Desktop Connector are already installed. Refer to *TP Desktop Connector for ACMS Installation Guide* for installation instructions.

Pre-requisites

Building, installing and executing the Conference Booking Appl sample requires some mandatory components to be there in both the OpenVMS as well as the Windows environment.

On the OpenVMS platform:

- Oracle RDB
- Oracle CDD
- HP ACMS
- HP TP Desktop Connector (OpenVMS component)

On the Windows platform:

- The Microsoft .NET Framework
- Visual Studio .NET
- HP TP Desktop Connector (Windows Component)

Procedure

1. Before building the sample application, copy the saveset file created by the HP TPware .NET installation to a directory where you will build the CONFERENCE_BOOKING_APPL application.
2. Extract the saveset using the command $ backup CONFERENCE_SAMPLE.SAV/sav [...]/log.
   The command extracts the files in the saveset to the sub-directory CONFERENCE_SAMPLE in the current directory.
3. Then, use the command $ @BUILD_CONFERENCE_SAMPLE
   The command does the following:
   - Creates the CDD directory and dictionary
   - Defines ACMS tasks, group and application
   - Builds the server code
   - Installs the ACMS application

The CDD definitions are created in the sub-directory CDD under the parent directory CONFERENCE_SAMPLE.

Starting the ACMS Application

Pre-requisites

Before starting the ACMS application, ensure the following:

- That the System Administrator has granted you the appropriate privileges
- That the TP Desktop Connector Gateway process (ACMSDI$GATEWAY) is started. Refer to *TP Desktop Connector for ACMS Gateway Management Guide* for detailed instructions.

Procedure

- To start the ACMS application, use the ACMS START APPLICATION command:
  $ ACMS/START APPLICATION CONFERENCE_BOOKING_APPL

  The following ensure that the command execution is successful:
  - The ACMS server application (CONFERENCE_BOOKING_APPL) being operational. It can be verified using the command ACMS/SHOW APPLICATION CONFERENCE_BOOKING_APPL
  - A gateway process listening to task requests, which can be verified using the command:
    UCX SHOW DEVICE_SOCKET/PORT=nnnn
    Note: The default port number is1023. Refer to the documentation to change the port number.

Once the preceding verifications are completed, you can start *Building the CONFERENCE BOOKING APPL Desktop Application*.

Building the CONFERENCE_BOOKING_APPL Desktop Application

You build the sample applications on the Windows 2000/XP/2003 system where the HP TP Desktop Connector software is already installed. Make sure that all of the pre-requisites mentioned for the Windows platform are already installed.

Pre-requisites

Before proceeding to building the desktop (C# .NET or VB .NET) applications, ensure that you complete the following tasks:

1. Copy the following Conference Booking sample self extracting archive files Conference_Booking_Sample_C#.exe and Conference_Booking_Sample_VB.exe from the installation directory to the directory where you will execute the build process.
2. Extract the files from these archives by executing them from the Command Prompt or use Windows Explorer to navigate to the folder where these files are located. The files from these archives will be extracted into the respective sub-directories with the same name as the archive.

Procedure

- If you are building the Visual C# .NET sample, proceed to *Building the CONFERENCE_BOOKING_APPL C# .NET Application*.
- If you are building the Visual Basic .NET sample, proceed to *Building the CONFERENCE_BOOKING_APPL Visual Basic .NET Application*.

Building the CONFERENCE_BOOKING_APPL C# .NET Application

Follow these steps to building the C#.NET application:

1. Open the solution file Conference_booking_sample_C#.sln using Visual Studio .NET
   Note: Ensure that the PATH system variable contains the path of the file ACMSDI.DLL that is available with the TP Desktop Connector Kit.
2. Use the Build menu in Visual Studio .NET to build the conference booking sample application.
   After the sample application is built, you can now proceed to *Running the Applications*.

Building the CONFERENCE_BOOKING_APPL Visual Basic .NET Application

Follow these steps to building the Visual Basic .NET application:

1. Open the solution file Conference_booking_sample_VB.sln using Visual Studio .NET
   Note: Ensure that the PATH system variable contains the path of the file ACMSDI.DLL that is available with the TP Desktop Connector Kit.
2. Use the Build menu in Visual Studio .NET to build the conference booking sample application.
   After the sample application is built, you can now proceed to *Running the Applications*.

Running the Applications

Running the Visual C# .NET Sample

Follow one of these steps to run the Visual C# .NET sample:

- Select Start or Start Without Debugging from the Debug menu to execute application from Microsoft Visual Studio .NET environment.

  OR

- Open Windows Explorer and navigate to the bin\debug directory in the Project directory. Double-click Conference_Booking_Sample_C#.exe to execute the application.

Running the Visual Basic .NET Sample

Follow one of these steps to run the Visual Basic .NET sample:

- Select Start or Start Without Debugging from the Debug menu to execute application from Microsoft Visual Studio .NET environment.

  OR

- Open Windows Explorer and navigate to the bin\debug directory in the Project directory. Double-click Conference_Booking_Sample_VB.exe to execute the application.

The STUDENT_APPL Sample

The STUDENT_APPL server sample that runs on your OpenVMS host consists of a Booking Student Database and Statistics application. The application consists of seven ACMS tasks that provide the functionality for the following features:

- Adding/modifying/deleting student data
- Viewing existing student data
- Viewing statistical information on existing student data

The STUDENT_APPL server sample consists of the following components:

- A OpenVMS saveset file containing the files for the ACMS application definitions and source code
- A Self-Extracting exe archive file containing a Visual C# .NET Web application deployed on IIS that demonstrates calling tasks using TPware .NET
- A Self-Extracting exe archive file containing a Visual Basic .NET Web application deployed on IIS that demonstrates calling tasks using TPware .NET

To begin with, you should build the STUDENT_APPL server sample. When the server application is completely built, you can proceed to building the STUDENT_APPL web applications and then you can execute the applications. Following is the sequence in which the sample applications are built and executed:

1. Building, Installing and Running the STUDENT_APPL Server
2. Starting the ACMS Application
3. Building the STUDENT_APPL Web Application
4. Building the STUDENT_APPL C# .NET Web Application
5. Building the STUDENT_APPL Visual Basic .NET Web Application
6. Installing and Running the Applications

Building, Installing and Running the STUDENT_APPL Server

The STUDENT_APPL server is built on the OpenVMS system.

Note: Ensure that an ACMS development system is already configured and OpenVMS components of the HP TP Desktop Connector are already installed. Refer to *TP Desktop Connector for ACMS Installation Guide* for installation instructions.

Pre-requisites

Building, installing and executing the STUDENT_APPL server sample requires some mandatory components to be there in both the OpenVMS as well as the Windows environment.

On the OpenVMS platform:

- Oracle RDB
- Oracle CDD
- HP ACMS
- HP TP Web Connector (OpenVMS component)

On the Windows platform:

- The Microsoft .NET Framework
- Microsoft Internet Information Server (IIS)
- HP TP Web Connector (Windows Component)

Procedure

1. Before building the sample application, copy the saveset file created by the HP TPware .NET installation to a directory where you will build the STUDENT_APPL application. Then extract the saveset using the command:
   $ backup STUDENT_SAMPLE.SAV/sav [...] /log
   The command extracts the files in the saveset to the sub-directory STUDENT_SAMPLE in the current directory.
2. Then, use the command $ @BUILD_STUDENT_SAMPLE
   The command does the following:
   - Creates the CDD directory and dictionary
   - Defines ACMS tasks, group and application
   - Builds the server code
   - Installs the ACMS application

The CDD definitions are created in the sub-directory CDD under the parent directory STUDENT_SAMPLE.

Starting the ACMS Application

Pre-requisites

Before starting the ACMS application, ensure the following:

- That the System Administrator has granted you the appropriate privileges
- That the TP Web Connector Gateway process (ACMSDA$GATEWAY) is started. Refer to TP Web Connector for ACMS Gateway *Management Guide* for detailed instructions.

Procedure

- To start the ACMS application, use the ACMS START APPLICATION command:
  $ ACMS/START APPLICATION STUDENT_APPL

  The following ensure that the command execution is successful:
  - The ACMS server application (STUDENT_APPL) being operational. It can be verified using the command ACMS/SHOW APPLICATION STUDENT_APPL

- A gateway process listening to task requests, which can be verified using the command UCX SHOW DEVICE_SOCKET/PORT=nnnn
  Note: The default port number is 1022. Refer to the documentation to change the port number.

Once the preceding verifications are done, you can start *Building the STUDENT APPL Web Applications*.

Building the STUDENT_APPL Web Application

You build the sample applications on the Windows 2000/XP/2003 system where the HP TP Desktop Connector software is already installed. Make sure that all of the pre-requisites mentioned for the Windows platform are already installed.

Pre-requisites

Before proceeding to building the web applications, ensure that you complete the following tasks:

1. Copy the following Student sample archive files Student_Sample_C#.exe and Student_Sample_VB.exe from the installation directory to the directory where you will execute the build process.
2. Extract the files from these archives by executing them from the Command Prompt or use Windows Explorer to navigate to the folder where these files are located. The files from these archives will be extracted into the respective sub-directories with the same name as the archive.

Procedure

- If you are building the Visual C# .NET sample, proceed to *Building the STUDENT_APPL C# .NET Web Application*.
- If you are building the Visual Basic .NET sample, proceed to *Building the STUDENT_APPL Visual Basic .NET Web Application*.

Building the STUDENT_APPL C# .NET Web Application

Pre-requisites

- Ensure that the .NET framework directory is in the system PATH variable. This directory is located either in the C:\Windows\Microsoft.NET\Framework or C:\Winnt\Microsoft.NET\Framework.
- The framework folder name depends on the version of .NET framework installed. It will be similar to v1.0.3705 or v1.1.4322. This folder contains the C# .NET and VB .NET compilers that is csc.exe and vbc.exe respectively.

Procedure

Follow these steps to building the C# .NET application:

1. Run the batch file Build_ STUDENT_APPL_C#.bat in the same folder where the Student Sample C# archive was extracted. Use the Command Prompt to run the batch file or use Windows Explorer to navigate to the folder where the file is located and then execute it.
   Note: If the build is executed from the Command Prompt, ensure that the directory used is the same as the one used for extracting the archive files.
2. After the build process is completed, the DLL file for the Student Sample C# application Student_Sample_C#.dll will be generated in a sub-folder called bin.

After the DLL file is generated, you can now proceed to *Installing and Running the Applications*.

Building the STUDENT_APPL Visual Basic .NET Web Application

Pre-requisites

- Ensure that the .NET framework directory is in the system PATH variable. This directory is located either in the C:\Windows\Microsoft.NET\Framework or C:\Winnt\Microsoft.NET\Framework.
- The framework folder name depends on the version of .NET framework installed. It will be similar to v1.0.3705 or v1.1.4322. This folder contains the C# .NET and VB .NET compilers that is csc.exe and vbc.exe respectively.

Procedure

Follow these steps to building the C# .NET application:

1. Run the batch file Build_ STUDENT_APPL_VB.bat in the same folder where the Student Sample VB archive was extracted. Use the Command Prompt to run the batch file or use Windows Explorer to navigate to the folder where the file is located and then execute it.
   Note: If the build is executed from the Command Prompt, ensure that the directory used is the same as the one used for extracting the archive files.
2. After the build process is completed, the DLL file for the Student Sample VB application Student_Sample_VB.dll will be generated in a sub-folder called bin.

After the DLL file is generated, you can proceed to *Installing and Running the Applications*.

Installing and Running the Applications

Installing the Applications

Follow these steps to install the sample applications:

1. Open the IIS Console. To do this, click Start, point to Settings and select Control Panel from the menu displayed.
2. In the Control Panel window, click the Administrative Tools icon.
3. On the window displayed, select the Internet Information Services icon.
4. Add a virtual directory to point to the respective web sample directory. For example, create a web virtual directory Student Sample C# that points to <disk>:\<Student Sample C#-directory>
   Note: Ensure that the Microsoft IIS is started before adding the virtual directory.

   This creates a web directory name that users will see from their web browsers. The web directory relates to the local Windows directory that contains your web server applications. Make sure that you enable read, script (if available), and execute permissions for the directory.
   Note: The method for creating a virtual directory depends on the specific version of IIS that is installed. Refer to the *Microsoft Documentation* for specific instructions.

Running the Applications

Follow these steps to run the sample applications:

1. Open the browser that is installed on your PC.
2. In the browser's Address box, browse to the location where the local IIS server and the web virtual directory created at the time of web pages installation are available. For example, if you set up a web directory named Student Sample C#, select http://localhost/Student Sample C#/login.aspx
3. The application starts by displaying the Sign-On page that prompts you to enter the following server access information:
   - Node name
   - User Name
   - Password

Once the information provided is authenticated by the sample application, you can proceed to perform any of the intended functions.

Migrating to TPware .NET

Introduction

If you are using TPware Desktop Connector/Web Connector, then the following generic migration procedure helps you to migrate from a Windows based TPware application, written using the TPware library, to a TPware .NET application using the .NET framework.

Advantages

Applications using the .NET framework have a number of advantages over native Windows applications. Some of the advantages are:

- Easier development of user interfaces
- More effective use of the business logic due to the availability of a wide variety of class libraries
- More secure and robust code due to the inherent properties of the .NET Application development framework
- Support for a wide range of languages

Migration Procedure

The TPware applications on Windows are used to execute the ACMS tasks running on OpenVMS. The tasks are executed by collecting the required data from the front-end interfaces. These tasks are then passed to the TPware Gateway server in the form of Workspaces. Data in the form of Workspaces is passed using the user-defined structures that contain the format for the data to be passed.

The major challenge in the migration of these structures to the .NET framework involves the mapping and marshalling of individual data elements in the structures. The following table lists the various native Windows data types and the corresponding .NET data types along with the related marshalling guidelines:

Table 1.13: Marshalling Guidelines

.NET CLR type	Unmanaged 'C' language type	Marshalling Scheme from Managed (.NET) to Unmanaged Space
Int16	Short	Default (No marshalling guideline)
U Int16	unsigned Short	Default (No marshalling guideline)
Int32	int, long	Default (No marshalling guideline)
U Int32	Unsigned int/unsigned long	Default (No marshalling guideline)
Boolean	Int/unsigned int	Default (No marshalling guideline)
Char	Char	Default (No marshalling guideline)
String	Fixed length char array char <variable> [<size>]	`[MarshalAs(UnmanagedType.ByValTStr,SizeConst= <size>)]`
Single	Float	Default (No marshalling guideline)
Double	Double	Default (No marshalling guideline)

Points to Remember During the Migration Process

- Windows floating point formats are based on the IEEE floating point formats. Hence, the ACMS applications on OpenVMS using floating-point numbers must use the IEEE formats for compatibility with Windows.
- The IEEE S_FLOAT format must be used for the single precision floating-point numbers and the IEEE D_FLOAT format must be used for double precision floating point numbers.
- ACMS applications must be compiled with the IEEE_FLOAT as the floating-point format.
- Applications should be linked with the VAXCRTLT library on the alpha architecture.
- The F_FLOAT (Single Precision) or D_FLOAT (Double Precision) floating point types must be used for all floating point CDD definitions.

Note:

- IEEE floating-point formats are only supported on the Alpha and the I64 architectures.
- IEEE float is the default floating point format on I64.

C Structure and Equivalent C# .NET Structure

The following 'C' structure and its corresponding C# .NET structure are given for your reference:

'C' Structure

```
struct  Part_Data
{
        char            Part_Name[32];
        int             Part_Number;
        int             Available; // Interpreted as Boolean i.e. true=1 and false=0
        float           Part_cost;
        double          Total_Stock_Value;
}
```

Equivalent C# .NET Structure

```
struct PartData
{
        [MarshalAs(UnmanagedType.ByValTStr,SizeConst=32)]
        String    PartName;
        Int32             PartNumber;
        Boolean Available
        Single    PartCost;
        Double    TotalStockValue;
};
```

Note:

- For more information on developing TPware .NET applications, refer to the Programmer's Reference Section.
- Samples applications for TPware .NET are also available in the Sample TPware .NET Code section.

TPwareReturnValues

The following table provides the list of TPware return values for every call executed by the client application. Every TPware return value is given a unique field name and appropriate description.

The TPwareReturnValue for every call executed by the client application can be broadly classified as:

- Informational messages
- Error messages
- Poll messages

Informational Messages

The following value is grouped under Informational messages. Click the value to view its description and field name:

- 0

Error Messages

The following values are grouped under Error messages. Click each value to refer to its description and field name:

- -3001
- -3002
- -3003
- -3004
- -3005
- -3006
- -3007
- -3008
- -3009
- -3010
- -3011
- -3013
- -3014
- -3015
- -3016
- -3017
- -3018
- -3019
- -3020
- -3021
- -3022
- -3023
- -3024
- -3025
- -3026
- -3027
- -3028
- -3031
- -3032
- -3033
- -3043
- -3044
- -3045
- -3100
- -3101

Poll Messages

The following values are grouped under Poll messages. Click each value to refer to its description and field name:

- -3102
- -3103
- -3104
- -3105

Index