Pro-face

Human Machine Interface

Pro-face

Data-Sharing API Driver

User Manual

Preface

Thank you for purchasing Pro-face's Pro-Designer software.

The Data-Sharing API Driver has been developed to make it easier to use the Data-Sharing API—the program module for accessing Pro-Designer variables from a custom user application (hereafter referred to as Data-Sharing API application). The Data-Sharing API Driver is provided with Pro-Designer.

This manual explains the functional specifications of the Data-Sharing API Driver. When actually programming the Data-Sharing API application, use this document in combination with the sample source code.

< Important >

(1) The copyrights to all programs and manuals included in the "Pro-Designer" software (hereafter referred to as "this product") are reserved by the Digital Electronics Corporation. Digital grants the use of this product to its users as described in the "END-USER LICENSE AGREEMENT" documentation. Any actions violating the above-mentioned agreement is prohibited by both Japanese and foreign regulations.

(2) The contents of this manual have been thoroughly inspected. However, if you should find any errors or omissions in this manual, please inform your local representative of your findings.

(3) Regardless of article (2), the Digital Electronics Corporation shall not be held liable by the user for any damages, losses, or third party claims arising from the uses of this product.

(4) Differences may occur between the descriptions found in this manual and the actual functioning of this product. Therefore, the latest information on this product is provided in data files (i.e. Readme.txt files, etc.) and in separate documents. Please consult these sources as well as this manual prior to using the product.

(5) The specifications set out in this manual are for overseas products only. As a result, some differences may exist between the specifications given here and for those of the identical Japanese product.

(6) Even though the information contained in and displayed by this product may be related to intangible or intellectual properties of the Digital Electronics Corporation or third parties, the Digital Electronics Corporation shall not warrant or grant the use of said properties to any users and/or other third parties.

Table of Contents

Chapter 1 Overview

Data-Sharing API and the Data-Sharing API Driver operate as a process of the Data-Sharing API application.

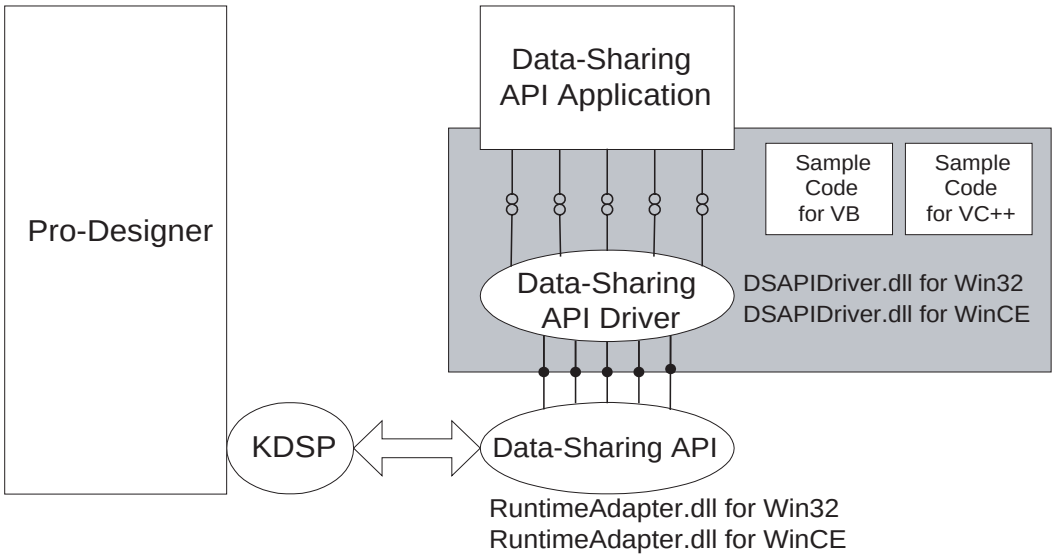

Diagram 1–1 Process/Module Image

You can create your Data-Sharing API application using the Data-Sharing API functions directly. However, the Data-Sharing API is designed for Visual C++® (VC++) users. Visual Basic (VB) users may find the structure somewhat complicated.

The Data-Sharing API Driver is designed to make the API structure as simple as possible, to meet the needs of users who want to program an application easily.

The Data-Sharing API Driver is used by loading the Data-Sharing API.

Any restrictions depend on the limitations of the Data-Sharing API.

Reference *For program limitations, refer to the* ***Data-Sharing API User Manual.***

Chapter 2 Structure

Functions provided by the Data-Sharing API Driver are divided into five major categories:

Initialize	Driver registers various values necessary for using the Data-Sharing API.
Open	Initializes the Data-Sharing API using the registered values.
Close	Runs the exit process.
Read	Using a variable handler, reads in the corresponding data.
Write	Using a variable handler, writes a value to the corresponding variable.

The process of an application using these functions:

Initialization ⇒ Open ⇒ {Read/Write} ⇒ Close

Appropriate functions differ according to the development environment (VB or Visual C++).

Reference *For information on the step-by-step processes in each case, refer to the sample source code.*

To use the functions in the Data-Sharing API Driver, you should understand the data structure. A simple explanation follows.

Target: A computer that runs Pro-Designer Runtime. Projects created in Pro-Designer, are downloaded to a target.

Setup requirements are as follows:

- Target IP Address
- Number of variables to be accessed
- Variable List

This information is defined by the structure data type: ST_DSAPI_TARGET.

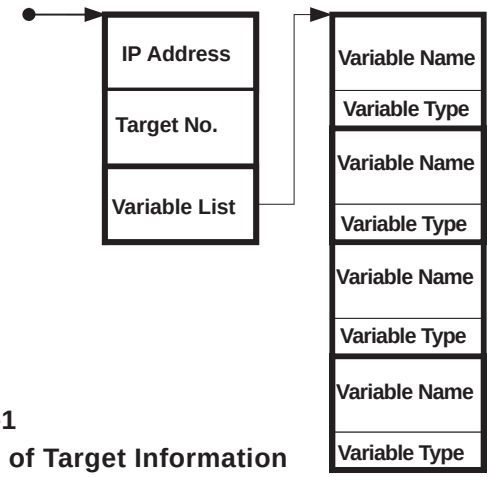

Diagram 2–1
Data Image of Target Information

Variable: Data storage area

The four basic variable data types are: Integer, Float, String, and Discrete.

Setup requirements are as follows:

- Variable name
- Data Type

This information is defined by the structure data type: ST_DSAPI_VARIABLE.

Environment Settings

When using the Data-Sharing API driver, the following files need to be present in the Application's environment.

Project.cfg: This is the Data-Sharing API setting file.

For details, refer to the Data-Sharing API User Manual.

An example (project.cfg) is provided in the Pro-Designer "Docs/CFG" folder. When you wish to use this file, simply copy it from this folder.

DSAPIDriver.dll: This is the Data-Sharing API driver.

This file is located in the Pro-Designer "Docs/Samples/DSAPIDriver/DLL" folder. Here, there are two versions - for Win32® and for Windows® CE. When required, copy the desired file to a folder in the Application's path.

Any machine used to run an application must also have either Pro-Designer Editor, Pro-Designer Runtime or Pro-Server Communication Tool installed.

When using Windows CE, if you copy the above files to a folder other than "Storage Card1", be sure to use [My Computer] -> [Control Panel] -> [Backup] to create a backup.

Chapter 3 Initialization / Open

This chapter explains the initialization process for the Data-Sharing API Driver. Among all the processes, initialization is the most complex.

Setup requirements are as follows:

- Environment setup (such as path)
- Target information (such as IP address)
- Registration of variables you want to reference

Specifically, the setting that enables you to access Pro-Designer variables is defined in the target properties. In other words, the variable list depends on the target information.

Differences in VB and VC++ Development

The basic process of an application developed using the Data-Sharing API Driver:

```
Initialization -> Open -> {Read/Write} -> Close
```

This process will not change, even if the Data-Sharing API application is developed in a VB or VC++ environment.

However, due to limitations in the VB language, VC or VC++ pointers to structure data types cannot be set up as a user-defined type in VB. As a result, you cannot use DSAPI_Int() to assign parameter values.

Rather than using DSAPI_Int(), VB is supported by the following initialization functions:

```
void    DSAPI_SetTarget(UINT32);
int     DSAPI_AddTarget(ST_DSAPI_TARGET*);
int     DSAPI_AddVariable(UINT32,ST_DSAPI_VARIABLE*);
void    DSAPI_Init_Ex(UCHAR*,UCHAR*,UCHAR*,UCHAR*);
```

Initializing the Data-Sharing API Driver

1. Define the number of targets using DSAPI_SetTarget().
2. Create target information: IP address and the number of variables set up.
3. Register the target information created in Step 2 using DSAPI_AddTarget().

   Targets are mapped with a number starting from zero (0), in the order of registration. These numbers are administered by the Data-Sharing API application.
4. Create variable information: names and data types.
5. Using the variable information created in Step 4, register the number of variables in each target using DSAPI_AddVariable().

The order of registered variables corresponds to the Index(variable_handler) used in Read or Write related functions.

Reference *For details about the order of registered variables, see the **Index(variable_handler)** section in Chapter 4.*

6. After targets and variables have been added, call DSAPI_Init_Ex() to set up the path to the development environment.
7. Call DSAPI_Open() to open the Data-Sharing API.

```
‘ Target number setup
  Call DSAPI_SetTarget(1)

‘ Target registration
  Call DSAPI_AddTarget(astTarget(0))
‘ Call DSAPI_AddTarget(astTarget(1))

‘ Variable registration
  Call DSAPI_AddVariable(0, astVarList1(0))
  Call DSAPI_AddVariable(0, astVarList1(1))
  Call DSAPI_AddVariable(0, astVarList1(2))
  Call DSAPI_AddVariable(0, astVarList1(3))

‘ Call DSAPI_AddVariable(1, astVarList2(0))
‘ Call DSAPI_AddVariable(1, astVarList2(1))
‘ Call DSAPI_AddVariable(1, astVarList2(2))
‘ Call DSAPI_AddVariable(1, astVarList2(3))

‘ Initialization of various setup items (for VB)
  Call DSAPI_Init_Ex(strLibPath, strCurrentPath, strSystemPath,
  strConfigPath)

‘ Open DSAPI
  Call DSAPI_Open
```

Changing the Registered Variables

DSAPI_Open can be executed only once in a single process.

If you wish to change more than once (2 or more times) the settings of the variables used for data transfer while an application is running, use DSAPI_Connect and DSAPI_Disconnect instead of DSAPI_Open and DSAPI_Close.

Using DSAPI_Connect and DSAPI_Disconnect

1. Call DSAPI_Disconnect.
2. Set the number of targets using DSAPI_SetTarget().
3. Create target information: the IP address and number of variables to set up.
4. Register the target information created in Step 3 using DSAPI_AddTarget().

   Targets are mapped with a number starting from zero (0), in the order of registration. These numbers are administered by the Data-Sharing API application.
5. Create variable information: names and data types.
6. Using the variable information created in Step 5, register the number of variables in each target using DSAPI_AddVariable().

   The order of registered variables corresponds to the Index(variable_handler) used in Read or Write related functions.

   Reference *For details about the order of registered variables, see the **Index(variable_handler)** section in Chapter 4.*
6. After targets and variables have been added, call DSAPI_Connect to reestablish the connection to the Data-Sharing API.

```
' Disconnect
  Call DSAPI_Disconnect

' Target number setup
  Call DSAPI_SetTarget(1)

'Target registration
  Call DSAPI_AddTarget(astTarget(0))
  Call DSAPI_AddTarget(astTarget(1))

'Variable registration
  Call DSAPI_AddVariable(0, astVarList1(0))
  Call DSAPI_AddVariable(0, astVarList1(1))
  Call DSAPI_AddVariable(0, astVarList1(2))
  Call DSAPI_AddVariable(0, astVarList1(3))

'Call DSAPI_AddVariable(1, astVarList2(0))
' Call DSAPI_AddVariable(1, astVarList2(1))
' Call DSAPI_AddVariable(1, astVarList2(2))
' Call DSAPI_AddVariable(1, astVarList2(3))

'Reconnect
  Call DSAPI_Connect
```

The When performing either reset or initialization, be sure to perform all steps in the order given.

Chapter 4 Read / Write Data

Index(variable_handler)

Index(variable_handler) must be used as the first parameter in each of the read and write functions listed below. This chapter explains how Index(variable_handler) is applied to variables.

```
DSAPI_Read(), ADSAPI_Write()
DSAPI_ReadInteger(), ADSAPI_ReadDiscrete(), ADSAPI_ReadFloat(),
DSAPI_ReadString(), ADSAPI_WriteInteger(), ADSAPI_WriteDiscrete(),
DSAPI_WriteFloat(), ADSAPI_WriteString()
```

Index(variable_handler) depends on the order variables were registered during initialization. If there is just one target, the order of registered variables is applied directly to Index(variable_handler) and corresponds to the handlers returned by the Data-Sharing API and Data-Sharing API Driver.

When an Integer variable is the first registered variable in the Data-Sharing API application, use the following to write data to this variable.

```
void*     pvData;
INT32     nData;

nData = 128;
pvData = (void*)&nData;       // Convert Integer variable to void type vari-
able.
pDSAPI_Write( 0, pvData );  // Write data to the variable in the 0 index
```

Variables are sequentially ordered. When multiple targets are registered, the first variable of the next target is the variable next to the previous target's last variable.

DSAPI_AddVariable()

When DSAPI_AddVariable() is used instead of DSAPI_Init(), it does not use the number of variables that had been registered in each target with DSAPI_AddTarget(). Instead, it uses as the upper limit, the number of variables registered using DSAPI_AddVariable().

Essentially, the first Index(variable_handler) of each target can be easily calculated by controlling the number of variables set up in each target.

Chapter 5 Data-Sharing API Driver I/F

The standard prefix used in the Dat-Shring API Driver is as follows:

Prefix	Category
DSAPI_	DSAPI Driver

DSAPIDriver.dll for Win32

```
void   DSAPI_Init(UINT32,ST_DSAPI_PATH,ST_DSAPI_TARGET*);
int    DSAPI_Open();
int    DSAPI_Close();
bool   DSAPI_Read(UINT32,void*,UINT16*);
bool   DSAPI_Write(UINT32,void*);
int    DSAPI_GetError();

int    DSAPI_Connect();
int    DSAPI_Disconnect();
int    DSAPI_Shutdown();

void   DSAPI_SetTarget(UINT32);
int    DSAPI_AddTarget(ST_DSAPI_TARGET_MB*);
int    DSAPI_AddVariable(UINT32,ST_DSAPI_VARIABLE_MB*);
void   DSAPI_Init_Ex(UCHAR*,UCHAR*,UCHAR*,UCHAR*);

int    DSAPI_ReadInteger(UINT32,INT32*);
int    DSAPI_ReadDiscrete(UINT32,UINT16*);
int    DSAPI_ReadFloat(UINT32,float*);
int    DSAPI_ReadString(UINT32,LPTSTR);
int    DSAPI_WriteInteger(UINT32,INT32*);
int    DSAPI_WriteDiscrete(UINT32,UINT16*);
int    DSAPI_WriteFloat(UINT32,float*);
int    DSAPI_WriteString(UINT32,LPCTSTR);
```

Name	Description	Return	Parameters
DSAPI_Init	Initializes the Data-Sharing API	void	UINT32, ST_DSAPI_PATH, ST_DSAPI_TARGET*
DSAPI_Open	Opens the Data-Sharing API	int	None
DSAPI_Close	Closes the Data-Sharing API	int	None
DSAPI_Read	Reads data	bool	UINT32, void*, UINT16*
DSAPI_Write	Writes data	bool	None
DSAPI_GetError	Gets errors	int	UINT32, void*
DSAPI_Connect	Connects variables	int	None
DSAPI_Disconnect	Cuts the connection	int	None
DSAPI_Shutdown	Deletes Data-Sharing API	int	None
DSAPI_SetTarget	Sets up a number of targets	void	UINT32
DSAPI_AddTarget	Registers target attributes	int	ST_DSAPI_TARGET_MB*
DSAPI_AddVariable	Registers variable attributes	int	UINT32, ST_DSAPI_VARIABLE_MB*
DSAPI_Init_Ex	Initialization function (VB)	void	UCHAR*, UCHAR*, UCHAR*, UCHAR*
DSAPI_ReadInteger	Reads Integer variables (VB)	int	UINT32, INT32*
DSAPI_ReadDiscrete	Reads Discrete variables (VB)	int	UINT32, UINT16*
DSAPI_ReadFloat	Reads Float variables (VB)	int	UINT32, float*
DSAPI_ReadString	Reads String variables (VB)	int	UINT32, LPTSTR
DSAPI_WriteInteger	Writes Integer variables (VB)	int	UINT32, INT32*
DSAPI_WriteDiscrete	Write Discrete variables (VB)	int	UINT32, UINT16*
DSAPI_WriteFloat	Writes Float variables (VB)	int	UINT32, float*
DSAPI_ WriteString	Writes String variables (VB)	int	UINT32, LPCTSTR

Name	:	**DSAPI_Init**
Category	:	**Win32**
Parameters	:	UINT32 nTargetCount, // Defines number of targets ST_DSAPI_PATH stPath, // Sets up structure necessary for initialization ST_DSAPI_TARGET* pstTargetList // Pointer to target information list
Return	:	void

Remarks	:	Registers the values for various settings for the Data-Sharing API.
		This function only registers values internally for the driver. The next step is to call DSAPI_Open(*)
		
		ST_DSAPI_PATH
		UNICHAR* puncLibPath // Path to RuntimeAdapter.dll
		UNICHAR* puncCurrentPath // Path to current execution directory
		UNICHAR* puncSystemPath // Path to system directory
		UNICHAR* puncConfigPath // Path to configuration file
		
		Example:
		C:\Program Files\pro-face\Docs\CFG\project.cfg
		
		ST_DSAPI_VARIABLE
		UNICHAR auncVarName[60] // Variable name
		BYTE byVarType // Variable data type
		VAR_TYPE_INT = 0,
		VAR_TYPE_FLOAT = 1,
		VAR_TYPE_STRING = 2,
		VAR_TYPE_DISCRETE = 3
		
		ST_DSAPI_TARGET
		UNICHAR* puncIPAddress // IP address of target machine
		UINT32 nVariableCount // Number of variables
		ST_DSAPI_VARIABLE* pstVariable // Pointer to variable list

Name	:	**DSAPI_Open**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Loads the Data-Sharing API library into memory and obtains the address for each method. Registers the function that is called when data is updated in the driver, or the variables to be shared. The variable list and other setup items are necessary when this function is called. Be sure to call DSAPI_Init() to set up all the necessary items before using DSAPI_Open().

Name	:	**DSAPI_Close**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Closes the Data-Sharing functions Releases the loaded library from memory

Name	:	**DSAPI_Read**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler void* pvData, // Returns the pointer to stored data UINT16* pnDataType // Returns data type
Return	:	**bool**
Remarks	:	Return value When data has changed: true When data has not changed: false **nIndex** Defines which variable in the variable list to read. **pvData** **pnDataType** When data gets updated, the pointer stored in this location identifies the data type.

Name	:	**DSAPI_Write**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler void* pvData // Pointer to the write data
Return	:	**bool**
Remarks	:	Return value If write operation is successful: true If write operation failed: false Defines a handler, and writes data to the corresponding variable.

Name	:	**DSAPI_GetError**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	When this function is called, returns error information. Because error information is overwritten every time an error occurs, this function always returns the latest error. RTA_CONNECTING = 0, RTA_CONNECTED = 1, RTA_TAGNAME_ERROR = 2, RTA_TOO_MANY_TAGS_ERROR = 3, RTA_VERSION_ERROR = 4

Name	:	**DSAPI_Connect**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Connects to the Data-Sharing API using the defined target and variable information.

Name	:	**DSAPI_Disconnect**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Disconnects from the Data-Sharing API and clears the target and variable information. To reconnect to the Data-Sharing API using DSAPI_Connect(), target and variable information must first be set up using DSAPI_SetTarget(), DSAPI_AddTarget(), and DSAPI_AddVariable().

Name	:	**DSAPI_Shutdown()**
Category	:	**Win32**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Ends the Data-Sharing API process. Typically, DSAPI_Close() is used to end the process. However, when DSAPI_Disconnect() is used to disconnect from the Data-Sharing API, this function must be used to end the process.

Name	:	**DSAPI_SetTarget**
Category	:	**Win32**
Parameters	:	UINT32 nTargetCount // The number of targets to register
Return	:	**void**
Remarks	:	When using DSAPI_Init_Ex(), use this function to register the number of targets.

Name	:	**DSAPI_AddTarget**
Category	:	**Win32**
Parameters	:	ST_DSAPI_TARGET_MB* pstTargetList // Pointer to the target // information list
Return	:	**int**
Remarks	:	Return value If operation is successful: true (not 0) If operation failed: false (0) Registers the target information, such as IP address and number of variables.

Name	:	**DSAPI_AddVariable**
Category	:	**Win32**
Parameters	:	UINT32 nTargetNum, // Defines the target ST_DSAPI_VARIABLE_MB* pstVariable // Pointer to the variable
Return	:	**int**
Remarks	:	Return value If operation is successful: true (not 0) If operation failed: false (0) Registers the variable name and type to the variable information in the defined target.

Name	:	**DSAPI_Init_Ex**
Category	:	**Win32**
Parameters	:	UCHAR* pucLibPath, // Path to RuntimeAdapter.dll UCHAR* pucCurrentPath, // Path to current execution directory UCHAR* pucSystemPath, // Path to system directory UCHAR* pucConfigPath // Path to configuration file
Return	:	**void**
Remarks	:	When DSAPI_Init() is not used, use this function instead. Prior to using the DSAPI_Open() function, use this function to register necessary information (DSAPI_SetTarget(), DSAPI_AddTarget(), and DSAPI_AddVariable()) to the Data-Sharing API driver.

Name	:	**DSAPI_ReadInteger**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler
		INT32* pnData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from Integer variables.

Name	:	**DSAPI_ReadDiscrete**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler
		UINT16* psData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from Discrete variables.

Name	:	**DSAPI_ReadFloat**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler
		float* pfData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from Float variables.

Name	:	**DSAPI_ReadString**
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler
		LPTSTR pucData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from String variables.

Name	:	DSAPI_WriteInteger
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler INT32* pnData // Pointer to 32bit data
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Use this function to write data to Integer variables.

Name	:	DSAPI_WriteDiscrete
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler UINT16* psData // Pointer to Boolean data
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Use this function to write data to Discrete variables.

Name	:	DSAPI_WriteFloat
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler float* pfData // Pointer to Float data
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Use this function to write data to Float variables.

Name	:	DSAPI_WriteString
Category	:	**Win32**
Parameters	:	UINT32 nIndex, // Data list handler LPCTSTR pucData // Pointer to String data
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Use this function to write to String variables.

DSAPIDriver.dll for WinCE

```
void   DSAPI_Init(UINT32,ST_DSAPI_PATH,ST_DSAPI_TARGET*);
int    DSAPI_Open();
int    DSAPI_Close();
bool   DSAPI_Read(UINT32,void*,UINT16*);
bool   DSAPI_Write(UINT32,void*);
int    DSAPI_GetError();

void   DSAPI_SetTarget(UINT32);
int    DSAPI_AddTarget(UNICHAR*,UINT32);
int    DSAPI_AddVariable(UINT32,UNICHAR*,BYTE);
void   DSAPI_Init_Ex(UNICHAR*,UNICHAR*,UNICHAR*,UNICHAR*);

int    DSAPI_ReadInteger(UINT32,INT32*);
int    DSAPI_ReadDiscrete(UINT32,UINT16*);
int    DSAPI_ReadFloat(UINT32,float*);
int    DSAPI_ReadString(UINT32,LPTSTR);
int    DSAPI_WriteInteger(UINT32,INT32*);
int    DSAPI_WriteDiscrete(UINT32,UINT16*);
int    DSAPI_WriteFloat(UINT32,float*);
int    DSAPI_WriteString(UINT32,LPCTSTR);
```

Name	Description	Return	Parameters
DSAPI_Init	Initializes Data-Sharing API	void	UINT32, ST_DSAPI_PATH, ST_DSAPI_TARGET*
DSAPI_Open	Opens Data-Sharing API	int	None
DSAPI_Close	Closes Data-Sharing API	int	None
DSAPI_Read	Reads data	bool	UINT32, void*, UINT16*
DSAPI_Write	Writes data	bool	UINT32, void*
DSAPI_GetError	Gets errors	int	None
DSAPI_SetTarget	Sets up a number of targets	void	UINT32
DSAPI_AddTarget	Registers target attributes	int	UNICHAR*, UNIT32
DSAPI_AddVariable	Registers variable attributes	int	UNIT32, UNICHAR*, BYTE
DSAPI_Init_Ex	Initialization function (VB)	void	UNICHAR*, UNICHAR*, UNICHAR*, UNICHAR*

Name	Description	Return	Parameters
DSAPI_ReadInteger	Reads Integer variables (VB)	int	UINT32, INT32*
DSAPI_ReadDiscrete	Reads Discrete variables (VB)	int	UINT32, UINT16*
DSAPI_ReadFloat	Reads Float variables (VB)	int	UINT32, float*
DSAPI_ReadString	Reads String variables (VB)	int	UINT32, LPTSTR
DSAPI_WriteInteger	Writes Integer variables (VB)	int	UINT32, INT32*
DSAPI_WriteDiscrete	Writes Discrete variables (VB)	int	UINT32, UINT16*
DSAPI_WriteFloat	Writes Float variables (VB)	int	UINT32, float*
DSAPI_ WriteString	Writes String variables (VB)	int	UINT32, LPCTSTR

Name : **DSAPI_Init**

Category : **WinCE**

Parameters : UINT32 nTargetCount, // Defines the number of targets
ST_DSAPI_PATH stPath, // Sets up the data structure data necesary for // initialization
ST_DSAPI_TARGET* pstTargetList // Pointer to target information list

Return : **void**

Remarks : Registers the values for various settings for the Data-Sharing API.

This function only registers values internally for the driver. The next step is to call DSAPI_Open(*)

ST_DSAPI_PATH

UNICHAR* puncLibPath // Path to RuntimeAdapter.dll
UNICHAR* puncCurrentPath // Path to current execution directory
UNICHAR* puncSystemPath // Path to system directory
UNICHAR* puncConfigPath // Path to configuration file

Example:
Storage Card1/project.cfg

ST_DSAPI_VARIABLE

UNICHAR auncVarName[60] // Variable name
BYTE byVarType // Variable data type
VAR_TYPE_INT = 0,
VAR_TYPE_FLOAT = 1,
VAR_TYPE_STRING = 2,
VAR_TYPE_DISCRETE = 3

ST_DSAPI_TARGET

UNICHAR* puncIPAddress // IP address of the target machine
UINT32 nVariableCount // Number of variables
ST_DSAPI_VARIABLE* pstVariable // Pointer to variable list

Name	:	**DSAPI_Open**
Category	:	**WinCE**
Parameters	:	None
Return	:	**int**
Remarks	:	Return value If operation is successful: true If operation failed: false Loads the DataSharing API library into memory and obtains the address for each method. Registers the function that is called when data is updated in the driver, or the variables to be shared. The variable list and other setup items are necessary when this function is called. Be sure to call DSAPI_Init() to set up all the necessary items before using DSAPI_Open().

Name	:	**DSAPI_Close**
Category	:	**WinCE**
Parameters	:	None
Return	:	**int**
Remarks	:	Returned value If operation is successful: true If operation failed: false Closes the Data-Sharing functions. Releases the loaded library from memory.

Name	:	**DSAPI_Read**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Defines the Index number of data in the data list void* pvData, // Returns the pointer to stored data UINT16* pnDataType // Returns data type
Return	:	**bool**
Remarks	:	Return value When data has changed: true When data has not changed: false **nIndex** Defines which variable in the variable list to read. **pvData** **pnDataType** When data gets updated, the pointer stored in this location identifies the data type.

Name	:	DSAPI_Write
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler void* pvData // Pointer to the write data
Return	:	**bool**
Remarks	:	Return value If operation is successful: true If operation failed: false Defines a handler, and writes data to the corresponding variable.

Name	:	DSAPI_GetError
Category	:	**WinCE**
Parameters	:	None
Return	:	**int**
Remarks	:	When this function is called, returns error information. Because error information is overwritten every time an error occurs, this function always returns the latest error. RTA_CONNECTING = 0, RTA_CONNECTED = 1, RTA_TAGNAME_ERROR = 2, RTA_TOO_MANY_TAGS_ERROR = 3, RTA_VERSION_ERROR = 4

Name	:	DSAPI_SetTarget
Category	:	**WinCE**
Parameters	:	UINT32 nTargetCount // The number of targets to register
Return	:	**void**
Remarks	:	When using DSAPI_Init_Ex(), use this function to register the number of targets.

Name	:	DSAPI_AddTarget
Category	:	**WinCE**
Parameters	:	UNICHAR* pucIPAddress, // IP address of the target machine UINT32 nVariableCount // Number of variables
Return	:	**int**
Remarks	:	Return value If operation is successful: true (not 0) If operation failed: false (0) Registers target information such as IP address and number of variables.

Name	:	**DSAPI_AddVariable**
Category	:	**WinCE**
Parameters	:	UINT32 nTargetNum, // Defines the target UNICHAR aucVarName[], // Variable name BYTE byVarType // Variable data type
Return	:	**int**
Remarks	:	Return value If operation is successful: true (not 0) If operation failed: false (0) Registers the variable name and type to the variable information in the defined target.

Name	:	**DSAPI_Init_Ex**
Category	:	**WinCE**
Parameters	:	UNICHAR* pucLibPath, // Path to RuntimeAdapter.dll UNICHAR* pucCurrentPath, // Path to current execution directory UNICHAR* pucSystemPath, // Path to system directory UNICHAR* pucConfigPath // Path to configuration file
Return	:	**void**
Remarks	:	When DSAPI_Init() is not used, use this function instead. Prior to using the DSAPI_Open() function, use this function to register necessary information (DSAPI_SetTarget(), DSAPI_AddTarget(), and DSAPI_AddVariable()) to the Data-Sharing API driver.

Name	:	**DSAPI_ReadInteger**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler INT32* pnData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value If data has changed: true If data has not changed: false Use this function to read in data from Integer variables.

Name	:	**DSAPI_ReadDiscrete**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler UINT16* psData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value If data has changed: true If data has not changed: false Use this function to read in data from Discrete variables.

Name	:	**DSAPI_ReadFloat**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		float* pfData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from Float variables.

Name	:	**DSAPI_ReadString**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		LPTSTR pucData // Returns the pointer to the stored data
Return	:	**int**
Remarks	:	Return value
		If data has changed: true
		If data has not changed: false
		Use this function to read in data from String variables.

Name	:	**DSAPI_WriteInteger**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		INT32* pnData // Pointer to 32bit data
Return	:	**int**
Remarks	:	Return value
		If operation is successful: true
		If operation failed: false
		Use this function to write data to Integer variables.

Name	:	**DSAPI_WriteDiscrete**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		UINT16* psData // Pointer to Boolean data
Return	:	**int**
Remarks	:	Return value
		If operation is successful: true
		If operation failed: false
		Use this function to write data to Discrete variables.

Name	:	**DSAPI_WriteFloat**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		float* pfData // Pointer to Float data
Return	:	**int**
Remarks	:	Return value
		If operation is successful: true
		If operation failed: false
		Use this function to write data to Float variables.

Name	:	**DSAPI_WriteString**
Category	:	**WinCE**
Parameters	:	UINT32 nIndex, // Data list handler
		LPCTSTR pucData // Pointer to String data
Return	:	**int**
Remarks	:	Return value
		If operation is successful: true
		If operation failed: false
		Use this function to write data to String variables.

Chapter 6 Restrictions

Because the Data-Sharing API Driver uses the Data-Sharing API, restrictions depend on the limitations in the Data-Sharing API. This chapter explains the main restrictions in the Data-Sharing API Driver.

Reference *For detailed information, refer to the **Data-Sharing API User Manual.***

Restrictions of the Data-Sharing API Driver

- The maximum number of target machines that can be connected is 16.
- There is a limit to the number of variables that can be used for data sharing. The following table shows the standard maximum Access Count allowed for each type of target machine.

Target Machine	Access Count (Maximum)
Windows Compatible PC (PL Series)	400
PS-G	150
GP, PS-P	150
Factory Gateway	75

  However, the "maximum" value given here is not a design limit, rather a value given that considers the performance speed of data updates. These values ultimately will depend on the amount of screen data to be processed, as well as other factors.
- When using this Data-Sharing API Driver with VB, data types that are not supported by the VB programming language (such as unsigned Integers) cannot be supported.