DataImport Provider

Version 1.1.5

User's guide

January 24, 2023

[Remarks]

[Revision history]

Version	Date	Description
1.0.0	2016-06-27	First edition.
1.0.1	2016-09-02	Changed the way of data reading from variable acquisition to Event notification. Changed/ added each option setting. Added an Execute command. Supported the under subfolder, so its target can be read.
1.0.2	2018-1-30	Added the treatment of CSV File.
1.1.0	2018-05-11	Added "SleepTime" option to AddController.
1.1.1	2018-10-30	Memory leak was corrected.
1.1.2	2019-02-14	Added "Reset" command.
1.1.3	2019-04-22	File read process was corrected. "Reset" command was corrected.
1.1.4	2020-05-19	Processing correction when file reading fails.
	2020-12-01	Clerical corrections.
	2021-11-30	Add and corrections to the Outline of provider. Corrected typo.
1.1.5	2023-01-24	Fixed a bug that caused abnormal termination when a large number of files exist in the monitoring folder during provider initialization.

[Supported models]

Version	Date	Note

Contents

1. Introduction

This is a user’s guide of CAO provider that accesses to CSV File in a cross-sectoral manner and acquires the data.

Hereafter, this provider (CaoProvDataImport.dll) is called DataImport provider.

Chapter 2 describes the outline of DataImport provider and detailed information about variables.

2. Outline of provider

2.1. Outline

DataImport provider is CAO provider that accesses to CSV File, which is stored in the specified folder, in a cross-sectoral manner; and reads/ notifies the each file's data in sequence as OnMessage event notification.

The file format is DLL (Dynamic Link Library) and it is dynamically uploaded from the CAO engine. To use DataImport provider, you need to install ORiN2SDK or manually perform the registration based on the following information.

Table 2-1 DataImport provider

File name	CaoProvDataImport.dll
ProgID	CaoProv.DataImport
Registration	Regsvr32 CaoProvDataImport.dll
Deregistration	Regsvr32 /u CaoProvDataImport.dll

2.1.1. Read Folder Operating Conditions

DataImport providers have been created assuming that you want to import folders that have the following actions: The figure below shows the applications running providers as ORiN applications and other applications.

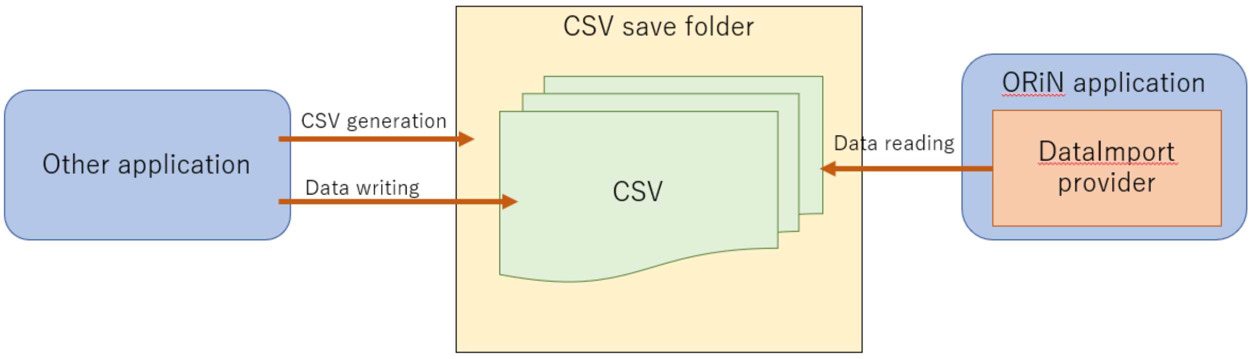

Fig. 2-1 Summary of Operation Assumed by DataImport Providers

In addition, the app dynamically creates a CSV file and adds data to the created CSV file. ORiN app monitors the CSV save folder and reads the data existing in the CSV file.

The following defines the behavior of other apps that DataImport providers expect.

Behavior of other applications

1 The CSV file is created dynamically. The file name is determined by the rules. (for example, a prefix is added, a file name is a date, and so on)

2 Data is added dynamically to the CSV file.

3 The CSV file to which data is added is always one in the created files.

4 Stopping the dynamic addition of data is stopped by rules in other applications (e.g. by adding 100 records)

5 No further data is added to the CSV file for which data addition is stopped.

6 When adding data to a file stops, a new CSV file is created. After that, the data is added to the created CSV file.

7 The CSV file in the folder is not moved, deleted, or renamed.

8 The data in the CSV file in the folder is not deleted.

The following figure shows one example of the above behavior.

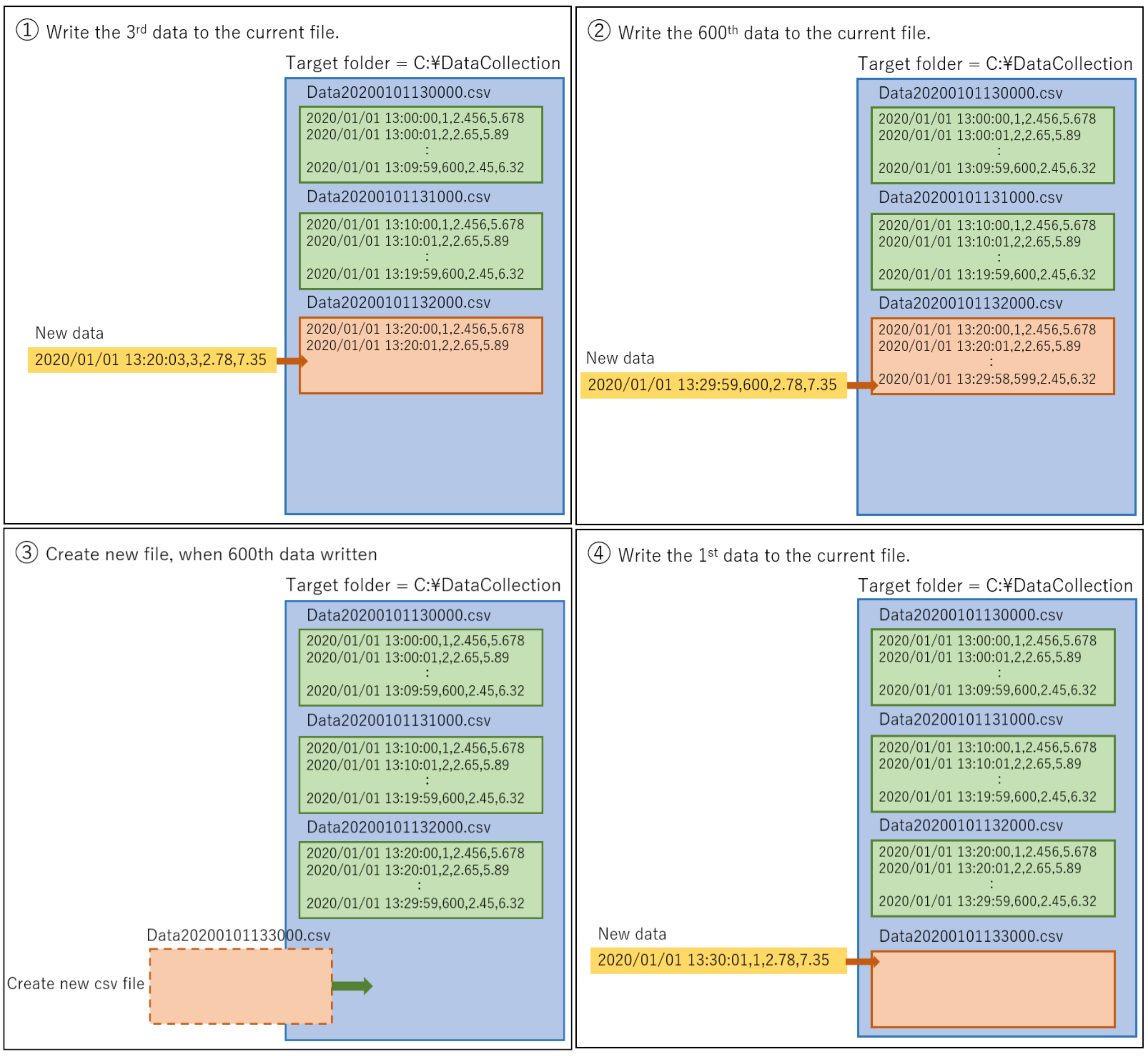

Fig. 2-2 How folders monitored by DataImport providers work

An application writes data to the most recent CSV file every second. When the application writes 600 data to a CSV file, it completes writing the data to the file and creates the next file. The filename will be "Data"+"yyyyMMddHHmmss". In the figure, the file to be written is indicated by 🟧 and the file to be written is indicated by 🟩.

2.1.2. Basic behavior of the provider

At the start of operation

When DataImport provider starts operation, it creates a list of files to be read that exist in the specified read-in folder (called a read-out file list). Determines the file to be read based on the read file list. For details on the read file list, see "2.1.2.1 Read file list"

At the end of operation

The interrupt data that summarizes the current read status is saved as a file, and processing is restarted based on the interrupted data at the next execution. For details of the interrupt data, see "2.1.2.3 Interrupt data file".

The figure below is a flowchart that summarizes the basic operation of DataImport providers.

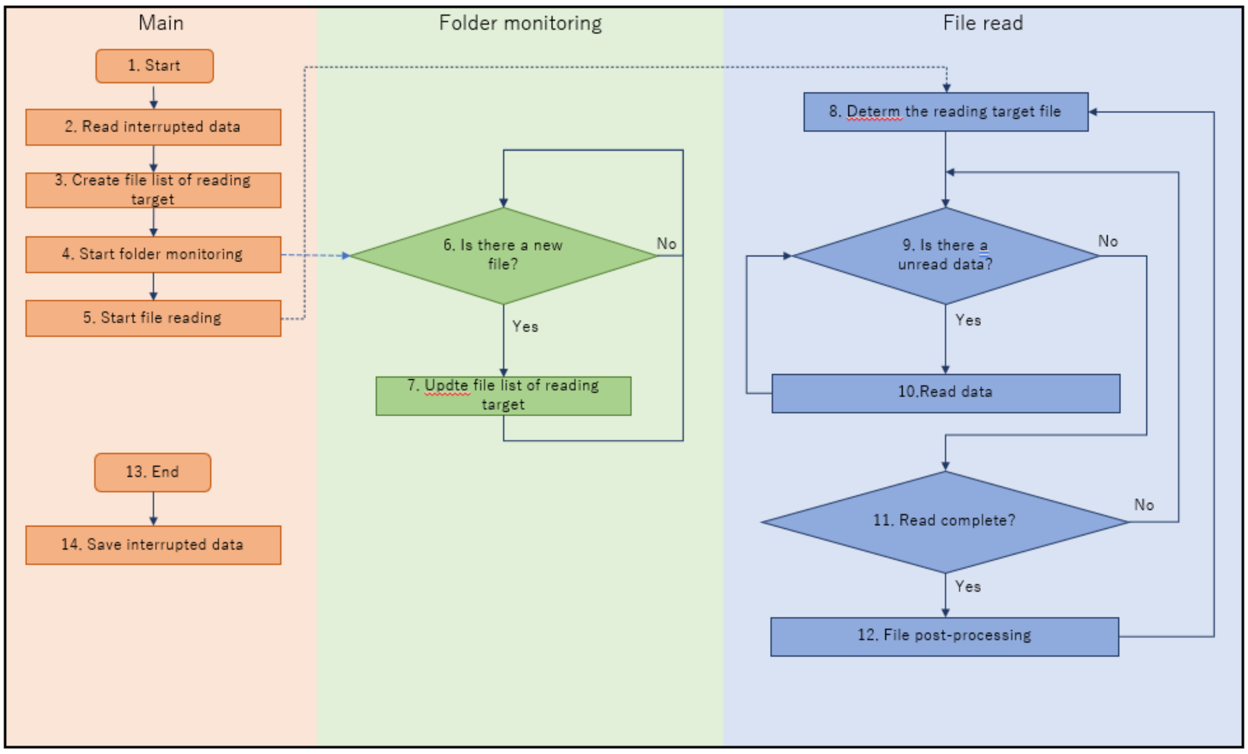
Main
Folder monitoring
File read
1. Start
2. Read interrupted data
3. Create file list of reading target
4. Start folder monitoring
5. Start file reading
13. End
14. Save interrupted data
6. Is there a new file?
No
Yes
7. Updte file list of reading target
8. Determ the reading target file
9. Is there a unread data?
No
Yes
10.Read data
11. Read complete?
No
Yes
12. File post-processing

Fig. 2-3 Summary of providers

A solid line indicates a synchronous call, and a dotted line indicates an asynchronous call.

Table 2-2 Outline of process

No	Type	Process name	Description
1	Event	Start	Indicates the provider's start of processing event. The call to "CaoWorkspace::AddController" is applicable.

No	Type	Process name	Description
2	Process	Read interrupted data	Reads the interrupt data saved at the last termination. For the interrupt data, refer to section "2.1.2.3 Interrupt data file".
3	Process	Create file list of reading target	Adds the interrupt data and creates a file list to be read. DataImport provider performs processing based on the file list created here. For details about the file read order, see "2.1.2.1.1 File Reading Order".
4	Process	Start folder monitoring	Starts monitoring of the read target folder specified when the controller was created. Folder monitoring is performed asynchronously.
5	Process	Start file reading	Starts file read processing. The file is read asynchronously.
6	Judgment	Is there a new file?	The Folder Monitor thread monitors whether a new file exists in the read-in folder. For details about the detection method, see "2.1.2.1.2 Detecting new read data (updating the file list)".
7	Process	Update file list of reading target	When a new file is detected, the detected file is added to the end of the read file list. For details about how to update the file list, see "2.1.2.1.2 Detecting new read data (updating the file list)".
8	Process	Determ the reading target file	Read from the beginning of the read file list.
9	Judgment	Is there a unread data?	Monitors whether there is unread data in the file. If there is unread data, proceed to "Data read processing". If there is no unread data, proceed to "There is an unread file" judgment.
10	Process	Read data	Reads one line of data from the file and returns to the "unread data" judgment.
11	Process	Read complete?	Determines whether reading is completed. When reading is completed, proceed to "File post-processing". When the read operation is not completed, the judgment returns to "unread data". For the judgment criteria, see "2.1.2.2 Read completion judgment".

No	Type	Process name	Description
12	Process	File post-processing	Performs post-processing of the file in the manner specified when the controller was generated. For more information about post-file processing, see "2.2.1.2 Types of file post-processing and how to specify them". When the post-file processing is completed, the unit returns to "Read file determination".
13	Event	Exit	The provider's end-of-process event. This is a call to CaoWorkspace::RemoveController method.
14	Process	Save interrupt data	Saves the current read status as interrupt data. For details, see "2.1.2.3 Interrupt data file".

2.1.2.1. Read file list

2.1.2.1.1. File Reading Order

Read file lists are filtered and sorted according to the options specified when the controller is generated. When the data read state reaches the start state, the file starts to be read from the beginning of this file list.

Note

- If a file name is specified for the sort method when a subfolder is selected, the list order is determined by the name of the file to which the path relative to the folder to be read is assigned.

2.1.2.1.2. Detecting new read data (updating the file list)

After the read file list is generated, the read target folder is monitored and new read data is detected. The target is the change in file size and update time in the target folder.

When a new file to be read is detected, it is added to the end of the read file list (the end of the read order) regardless of the sort setting. If multiple additional candidates are found at the same time, they are added to the file list in the order specified in the sort.

2.1.2.2. Read completion judgment

If another unread file exists when reading to the end of the current file is completed, reading of the current file is completed and the post-processing is executed.

If there is no unread file in the read file list even if reading to the end of the current file is completed, data may be added to the current file. Therefore, post-processing is not executed and file reading continues.

After the post-processing is performed, the file list is also updated to the post-processing state. Therefore, if a new file with the same name as before and after processing is added, it is regarded as a new file.

2.1.2.3. Interrupt data file

2.1.2.3.1. File Configuration

The following contents are described in the interrupt data file.

Table 2-3 Instruction of file of interrupted data record

Section name	Key name	Description
LastFile	Path	Folder path of the reading target
	FullPath	Full path of final reading
	Name	File name of final reading
	Line	Last read line count
	Elem	Number of data elements for the line of final reading
FileFilter	NameFilter	File name of Filter
TimeFilter	Begin	Update time filter (Absolute time: begin)
	End	Update time filter (Relative time: end)
	BeginRelative	Update time filter (Relative time: begin)
	EndRelative	Update time filter (Relative time: end)
Sort	Type	Sort type

2.1.2.3.2. Record

When the controller is released, a file (controller name + history.ini) is generated that records the file information being read, various filter conditions, and sorting conditions as interrupt data. The location of the generation is optionally specified in CaoConfig parameter settings under "HistoryPath". If not specified, it will be generated in the same folder as the provider's DLL.

2.1.2.3.3. Resume from suspended data

The next time the controller with the same name is generated, if this interrupt data recording file exists and the read conditions (various filters and sorts) are the same and the target file exists, the read operation starts from the previous one. If the read conditions are different or the target file does not exist, read data from the beginning of the read file list created when the controller was created.

Note

- Even if the status of the read-target folder is restored to the initial state at the time of restart, it is determined that the read operation has been completed and may not be read.
- If the controller name is not specified, the interrupt data cannot be used because the controller name is random.

2.1.2.3.4. Reset Suspended Data

Suspended data can also be reset in progress. Use Reset command to reset it.Reset command

To leave the interrupted data correctly, when deleting the controller object, be sure to change the state of reading to the state of stop reading and delete it. Use "StopReading command" **to bring it down.**

If deleting the controller object in the state of start reading, its reading processing will be proceeded during deleting the object and the interrupted data remains as a result. (Because the reading data processing starts at the point where the reading processing to resume has proceeded, some data is missing.)

2.2. Method and Properties

2.2.1. CaoWorkspace::AddController method

DataImport provider references the parameters on the filter conditions when AddController; and creates the file list targeted of reading, under the specified subfolders.

When deleting controller, executing StopReading command shall be required. (See: 2.2.4.2)

Syntax AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

bstrCtrlName	: [in] Controller name
bstrProvName	: [in] Provider name. Fixed to =” CaoProv.DataImport”
bstrPcName	: [in] Computer name where provider runs.
bstrOption	: [in] Option character strings

The following table shows a list of option character strings.

Table 2-1 Option character strings of CaoWorkspace::AddController

Option ([1])	Description
Path=<Folder path>	Specify a destination folder of the reading target file. The folder path should be an absolute path. (Required)
NameFilter[=<File name filter>]	Specify a File name filter of the reading target. The filter can be specified with the wildcard. (* and ? can be used) (Default: without filter)
TimeFilterBegin[=<Update time of target file>]	Specify the update time filter (specify absolute time: start time) of the reading target file. (See: 2.2.1.1)
TimeFilterEnd[=<Update time of target file>]	Specify the update time filter (specify absolute time: end time) of the reading target file. (See: 2.2.1.1)
TimeFilterBeginRelative[=<Update time of target file>]	Specify the update time filter (specify relative time: start time) of the reading target file. (See: 2.2.1.1)
TimeFilterEndRelative[=<Update time of	Specify the update time filter (specify relative time: end

[1] Parameters surrounded by the square brackets (“[ ]”) can be omitted. Underlined part shows the default value when the option is not specified.

target file >]	time) of the reading target file. (See: 2.2.1.1)
Sort[=<Sort type>]	Specify the reading order of the reading target file. (Default: 3) <table><tr><th>Value</th><th>Sort type</th></tr><tr><td>1</td><td>File name ascending order</td></tr><tr><td>2</td><td>File name descending order</td></tr><tr><td>3</td><td>Update time ascending order</td></tr><tr><td>4</td><td>Update time descending order</td></tr></table>
PostProc[=<File post-processing type >]	Specify the way of post-processing file that has finished reading. (Default: 1) <table><tr><th>Value</th><th>File post-processing type</th></tr><tr><td>1</td><td>Do nothing</td></tr><tr><td>2</td><td>Rename</td></tr><tr><td>3</td><td>Move</td></tr><tr><td>4</td><td>Delete</td></tr></table>(See:2.2.1.2)
PostProcParam[=<Parameter>]	Specify the parameters per way of post-processing file. (See:2.2.1.2)
CSVStartLine[=<Reading start line>]	Specify the line that starts reading per file. (Default: 1)
CSVDelimiter[=<Delimiter>]	Specify the data delimiter. (Default: 1) (See:2.2.1.3)
CSVQuotation[=TRUE/FALSE]	Specify whether interpreting is executed with the double quotation marks surrounding the data. (Default : FALSE) TRUE: Execute interpreting FALSE: Do not execute interpreting (See:2.2.1.4)
CodePage[=<Code page No.>]	Specify the character code of CSV file with the Code page No. (Default: 0) e.g.1) ANSI Code Page = 0 e.g.2) Shift_JIS = 932 e.g.3) UTF-8 = 65001
MessageMax[=<Maximum number of message]	Specify the object number that was created by the event notification of data reading.

	When the notification has reached the specified number, data reading processing is blocked until the message object is released and possible to notify again. (Default: 100)
EnableSubfolder[=TRUE/FALSE]	Specify whether under the subfolder in the specified folders by Path option get targeted as a reading data. (Default: FALSE) TRUE: get targeted FALSE: don’t get targeted
AutoStartDelay[=<Auto start Delay time>]	Specify to start reading processing some milliseconds later from AddController. (Default: 100) -1: Reading processing is Stop sate. Execute StartReading command; and start reading any time of the timing. More than 0: Start reading after designated milliseconds. When executing StartReading command before the time specified, its reading will be started instantly.
SleepTime[=<Sleep time>]	The stop time when the change of the watch folder is detected is set. (Default：1000)

2.2.1.1. Update time filter of files

The following shows the detailed option parameters to specify the update time filter of files.

Although to specify with a combination of these parameters is possible, when specifying with the absolute time in combination with relative time, the absolute time is prioritized.

When the option is not specified, it will be operated as none of the corresponding filter is set.

[Specify absolute time: Start time]

“TimeFilterBegin[=<Start time >]”

< Start time >	:	Specify the start time with the absolute time. A newer one than the file specified update time is targeted for reading. The following designates the form of year, month, date, hour, minute and second. Form: YYYY/MM/DD hh:mm:ss

e.g.) TimeFilterBegin=2016/01/15 12:05:30

- YMD are separated each with”/”.
- hhms are separated each with”:” (colon)
- A half size space is required for between YMD and hms.

[Specify absolute time: End time]

“TimeFilterEnd[=<End time >]”

< End time >	:	Specify the end time with the absolute time. An older one than the file specified update time is targeted for reading. The following designates the form of year, month, date, hour, minute and second. Form: YYYY/MM/DD hh:mm:ss

※ Details are the same as < Start time >.

[Specify relative time: Start time]

"TimeFilterBeginRelative[=<Start time >]"

< Start time >	:	Specify the start time with the relative time after executing AddController. Relative time can be specified by seconds by decimal number that is towards the past, positive direction. A newer one than the file specified update time is targeted for reading.

e.g. 1) TimeFilterBeginRelative=300

e.g. 2) TimeFilterBeginRelative=-300

- ➢ 300 seconds = the file, which has been updated later of the time of 5-minute ago of AddController, is targeted.
- ➢ If specified the negative value, it shows the future time.
- ➢ -300 seconds = the file, which has been updated later of the time of 5-minute later of AddController, is targeted.

[Specify relative time: End time]

"TimeFilterEndRelative[=<End time >]"

< End time >	:	Specify the end time with the relative time after executing AddController. Relative time can be specified by seconds by decimal number that is toward the past, positive direction. An older one than the file specified update time is targeted for reading.

e.g. 1) TimeFilterEndRelative=300

e.g. 2) TimeFilterEndRelative=-300

- ➢ 300 seconds = the file, which has been updated before the time of 5-minute ago of AddController, is targeted.
- ➢ If specified negative value, it shows the future time.
- ➢ -300 seconds = the file, which has been updated before the time of 5-minute later of AddController, is targeted.

■ Examples of specifying with combinations

e.g.1) Specify the start /end time with the absolute time

“TimeFilterBegin=2016/01/15 12:05:30, TimeFilterEnd=2016/01/16 00:00:00”

The file that was updated from 2016/01/15 12:05:30 to 2016/01/16 00:00:00 is targeted.

e.g.2) Specify the start /end time with the relative time

“TimeFilterBeginRelative=3600, TimeFilterEndRelative=300”

The file, which has been updated during the time from 3600- second (=1 hour) to 300- second (=5-minute) ago, is targeted.

(If the time of AddController is 13：00, the file that was updated from 12:00 to 12：55 is targeted.)

e.g.3) Specify the time with a combination of the absolute and relative times

“TimeFilterBegin=2016/01/15 12:05:30, TimeFilterEndRelative=300”

The specified time with the absolute time (with specified start time, without specified end time) is prioritized because TimeFilterBegin had been specified; the option with the specified relative time is disabled.

In this case, all the files that were updated after 2016/01/15 12:05:30 are targeted.

2.2.1.2. Types of file post-processing and how to specify them

The following shows the way of specifying per type of the file post-processing.

[Do nothing]

"PostProc=1"

After completing 1-file reading, do nothing to the file.

To specify PostProcParam option is not needed; if being specified, it will be ignored.

[Rename]

"PostProc=2, PostProcParam=< extension after changing>"

After completing 1-file reading, rename the file's extension name with the specified character strings at PostProcParam option.

¥ / : * ? "< > | cannot be included in the character strings.

If the characters are specified in which replacing the extension results in the incorrect file name, the post-processing will be failed.

If the specified characters result in an error as a file name when replacing the extension, its post-processing will fail.

If the file with the same name exists after renaming, its post-processing will fail. (Not overwritten)

[Move]

"PostProc=3, PostProcParam=< Destination folder path >"

After completing 1-file reading, move the file to the specified folder path at PostProcParam option.

The folder path should be specified with absolute path, as well as an existing folder should be specified. (Generating a folder automatically is impossible)

If the file with the same name exists in the destination folder, its post-processing will fail. (Not overwritten)

e.g.) File: c:¥foo¥hoge.csv, Destination folder path: c:¥bar

c:¥foo¥hoge.csv → c:¥bar¥hoge.csv

[Delete]

"PostProc=4"

After completing 1-file reading, delete the file.

To specify PostProcParam option is not needed; if being specified, it will be ignored.

2.2.1.3. Data delimiter

The following shows the way of specifying data delimiter.

"CSVDelimiter[=<Data delimiter >]"

Select one among data delimiter, numerical value and character to specify. (Default 1)

- If specify with numerical values

1	:	comma-delimited
2	:	tab-delimited

- If specify with characters

  Specify a character (1 character) with data delimiter, by surrounding it with double quotation marks.
  To specify double quotation marks as data delimiter is impossible.

[Examples of specifying]

e.g.1) Comma-delimited (Specify numerical values)
"CSVDelimiter=1"

e.g.2) Comma-delimited (Specify characters)
"CSVDelimiter=(",")"

e.g.3) Tab-delimited (Specify numerical values)
"CSVDelimiter=2"

e.g.4) Space-delimited (Specify characters)
"CSVDelimiter=(" ")"

e.g.5) Slash-delimited (Specify characters)
"CSVDelimiter=("/")"

2.2.1.4. Interpretation of the data surrounded with double quotation marks

The following shows the way of interpretation of the data surrounded by double quotation marks. ([2])

[In case of being seen as csv]

	1	2	3
1	AAAA	BBBB	CCCC
2	1111	22 33	4444
3	“ABCD”	EF”GH	55,66

[In case of being seen as txt]

1	AAAA, BBBB, CCCC
2	1111, “22
3	33”, 4444
4	“””ABCD”””, “EF””GH”, “55,66”

[Data acquisition]

<Do not execute interpretation with double quotation marks>

Line 1	Line 2	Line 3	Line 4
[0] AAAA	[0] 1111	[0] 33”	[0] “””ABCD”””
[1] BBBB	[1] “22	[1] 4444	[1] “EF””GH”
[2] CCCC			[2] “55
			[3] 66”

< Execute interpretation with double quotation marks>

Line 1	Line 2	Line 3
[0] AAAA	[0] 1111	[0] “ABCD”
[1] BBBB	[1] 22(¥r¥n)33	[1] EF”GH
[2] CCCC	[2] 4444	[2] 55,66

※ (¥r¥n) …. Line break code

[2] The equivalent interpretation is executed to tab-delimited; the only difference is its delimiter.

2.2.2. CaoController::AddVariable method

AddVariable method of CaoController class creates a variable object.

Syntax AddVariable(<bstrVariableName:VT_BSTR>[,<bstrOption:VT_BSTR>])

<bstrVariableName>	:	[in] Variable name
<bstrOption>	:	[in] Option character string

2.2.3. CaoVariable:get_Value property

Acquire values of variable object.

2.2.4. CaoController::Execute

Execute method of CaoController class executes commands.

Syntax [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

< bstrCmd>	:	[in] Command name
< vntParam>	:	[in] Parameter

The following list shows the commands available.

Table 2-2 List of Command

Command	Function	Parameter
StartReading	Start reading data	None
StopReading	Stop reading data	None
Reset	Reset of data reading position	< Auto start delay time > (VT_I4) Whether the read processing is begun from Reset milli how many second after is specified. -1: The read processing is completed by the halt condition. The StartReading command is executed, and reading of arbitrary timing is begun. 0 or more: After a specified millisecond,

		reading is begun. When the StartReading command is executed before specified time passes, reading is immediately begun.

2.2.4.1. StartReading command

This command changes the state of reading data to the state of start reading.

When the state of reading has turned out the state of start reading, the command reads 1- line data in sequential order, which is in accordance with each option designation from the target file, and issues the reading data notification (see:2.2.5.2) by reading OnMessage event.

When there is no reading target data or the number of message object has reached the maximum number specified at the option, reading processing and reading data notification will be interrupted and blocked. When the reading target data has been added or the message object is unable to be notified after releasing, the data reading processing and the reading data notification will start resuming automatically.

2.2.4.2. StopReading command

This command changes the state of reading data to the state of stop reading.

Executing StartReading command again enables the reading.

2.2.4.3. Reset command

The data reading position is reset.

It is possible to try to read it from the head again by disregarding the interruption data.

2.2.5. CaoController::OnMessage event

OnMessage event of CaoController class will be generated as shown in the table below.

Table 2-3 Message classification

Message classification		Cause of occurrences
0	Error log	When internal error is found, Log will be outputed at INFO level. (Data format error detection, File post-processing failed, etc.)
1	Notification of reading data	Notify 1-line data analyzed in accordance with each option designation, as dividing per data. When the message state of reading is turned out the state of start reading, a notification will be informed in sequential order. (When the message object has reached the specified number, data reading processing will be blocked.)
2	Notification of a change of data structure	When the data elements per 1-line is different from the previous data, a notification will be informed.

2.2.5.1. Error Log

Data types obtained by Error log messages are as follows:

Number	:	Message classification (0)
Value	:	Error contents (VT_BSTR)
DateTime	:	Time stamp
Destination	:	Null
Source	:	Null
Description	:	Null

2.2.5.2. Notification of reading data

Data types obtained by messages of Notification of reading data are as follows:

Number	:	Message classification (1)
Value	:	Data read out per 1-line (VT_BSTR | VT_ARRAY)
DateTime	:	Time stamp
Destination	:	Null

Source	:	File name ([3])
Description	:	Null

2.2.5.3. Notification of a change of data structure

Data types obtained by messages of Notification of a change of data structure are as follows:

Number	:	Message classification (2)
Value	:	New data elements (VT_I4)
DateTime	:	Time stamp
Destination	:	Null
Source	:	Null
Description	:	Null

2.3. Variable list

2.3.1. CaoController class

Table 2-4 CaoController class List of user variable

Variable name	Data type	Description	Attribute	
			get	put
—	—	Not available	—	—

Table 2-5 CaoController class List of system variable

Variable name	Data type	Description	Attribute	
			get	put
@VERSION	VT_BSTR	Provider version information	✓	—

2.4. Error code

There is no specific error code in DataImport Provider.

For ORiN2 common errors, refer to the section of Error code, “ORiN2 Programming Guide”.

[3] In case of the file under the subfolder, a relative path that was specified from the folder at Path Option will be given before the file name.