COLMINA プロバイダ
REST API 対応

Version 1.5.0

ユーザーズガイド

April 21, 2021

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2018-4-16	初版.
	2018-6-5	登録（JSON，CSV，TXT，BIN），最新データ参照，登録データ HIT 数取得コマンドを Variable として実装.
	2018-7-11	下記 API コマンドを Execute として実装. リソース制御 (登録，更新，削除) , アクセスコード制御 (登録，参照，更新，削除), イベント制御 (登録，参照，更新，削除)
	2018-7-20	REST 時のレスポンスステータスコード を追加
	2018-7-30	概要に API について簡易説明追記 filter 条件に例を追記, アクセスコード参照とイベントコード参照時に使用可能な filter 条件を追記
	2018-8-2	サンプルプログラム追加
	2018-8-31	サンプルプログラム追加(Variable，汎用 REST，データ制御，リソース制御，アクセスコード制御，イベント制御)
1.0.1	2019-3-7	「はじめに」の章を修正
1.1.0	2019-8-29	Extension を追加. プロキシ設定を追加. 不要なコマンド・変数を削除.
1.2.0	2019-10-8	Insecure オプション追加.
1.3.0	2019-10-23	・CSV データ登録 API ・JSON データ登録 API ・COLMINA エッジ IoT-PF API 対応.
1.4.0	2020-1-24	・データレイク API(NoSQL) ・データレイク API(RDB) 対応.
1.5.0	2021-4-21	アウトオブプロセスに変更.

目次

1. はじめに

COLMINA プロバイダは，富士通 「ものづくりデジタルプレイスCOLMINA」が提供する各種APIの呼び出しを提供するプロバイダです．

COLMINAは工場の設備や人・製造物の情報,ものづくり全般に関わる業務システムやノウハウの連携,さらに企業間でのサプライチェーン連携などを実現します．

このプロバイダを利用することで，生産・製造現場の様々な情報をCOLMINAが提供する各種サービスにつなぐことが可能となります．

本ドキュメントでは，COLMINA プロバイダの概要と，実装されているCAOインタフェース（関数仕様）について説明しています．

2. プロバイダの概要

2.1. インストール

COLMINA プロバイダモジュールは，下記の EXE で構成されています．ORiN2 SDK のインストーラでインストールした場合は，インストール作業は不要です．手動でインストールする場合は，表 2-1 のように実行してください．

表 2-1 COLMINA プロバイダ

ファイル名	CaoProv.FUJITSU.COLMINA.exe
ProgID	CaoProv.FUJITSU.COLMINA
レジストリ登録	RegistAsm.bat CaoProv.FUJITSU.COLMINA.exe
レジストリ登録の抹消	UnregistAsm.bat CaoProv.FUJITSU.COLMINA.exe

※ RegistAsm.bat および UnregistAsm.bat は{ORiN2 インストールフォルダ}¥DotNet¥Bat 下にあります．

2.2. 概要

COLMINA プロバイダは，COLMINA が提供する各種サービス活用のための API(汎用 REST， データ制御など)をラップした機能を提供します．

使用できる API には以下のものがあります．

・CSV データ登録 API

・JSON データ登録 API

・データレイク API（NoSQL）

※本文中の「非構造化データ」はデータレイク API（NoSQL）のデータを表します

・データレイク API（RDB）

※本文中の「構造化データ」はデータレイク API（RDB）のデータを表します

・COLMINA エッジ IoT-PF API

※ご利用いただく API 毎に設定項目が異なります．詳細は COLMINA の各 API リファレンスをご参照ください．

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

書式　AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, <bstrOption:BSTR>)

< bstrCtrlName > : [in] コントローラ名
< bstrProvName > : [in] プロバイダ名. 固定値 ="CaoProv.FUJITSU.COLMINA"
<bstrPcName> : [in] プロバイダの実行マシン名（未使用）
<bstrOption> : [in] オプション文字列="<オプション 1>,<オプション 2>,…"

- ベース URL
- タイムアウト
- プロキシサーバアドレス
- プロキシサーバポート番号
- プロキシサーバユーザ名
- プロキシサーバパスワード
- 証明書チェック設定

設定項目=設定内容の形で, カンマ区切りで指定します.
設定例:
"BaseURL=http://<zone>.fujitsu.com, Timeout=30000,…"

以下に, オプション文字列(bstrOption)の詳細を示します.

表 2-2 CaoWorkspace:: AddController メソッドのオプション文字列

設定項目	設定内容	必須	備考
BaseURL	http://<zone>.fujitsu.com	○	<zone>に入る値は, COLMINA のご契約後の通知内容に従ってください.
Timeout	<Timeout>	-	REST 通信時のタイムアウト時間をミリ秒単位で設定します. 省略時は, 30000 が設定されます.
ProxyServer	<Proxy server address>		プロキシサーバのアドレス. 省略時はシステム既定の設定を使用します.

			設定例: aaaa.bbbb.cccc.dddd
ProxyPort	<Proxy server port number>		プロキシサーバのポート番号. [ProxyServer]設定時に省略すると[80]が使用されます.
ProxyUserName	<Proxy server user name>		プロキシサーバのユーザー名.
ProxyPassword	<Proxy server password>		プロキシサーバのパスワード.
Insecure	<True/False>		セキュア通信時にサーバ証明書とサーバの整合性チェックの省略設定を指定します. True : チェックを省略する. False : チェックを省略しない.（デフォルト）

2.3.2. CaoController::AddExtension メソッド

書式　AddExtension (<bstrExtensionName:BSTR>, <bstrOption:BSTR>)

< bstrExtensionName > : [in] エクステンション名

<bstrOption> : [in] オプション文字列="<オプション 1>,<オプション 2>"

共通
- ・APIType
- ・API バージョン

CSV データ登録 API
- ・テナント ID
- ・ユーザー名
- ・パスワード

JSON データ登録 API・COLMINA エッジ IoT-PF API
- ・アクセスコード
- ・ポート番号

データレイク API（NoSQL）

データレイク API（RDB）
- ・テナント ID
- ・ユーザーID
- ・パスワード

・API キー

設定項目=設定内容の形で，カンマ区切りで指定します．
設定例：
"APIType=<APIType>, APIVersion=v1, …"

表 2-3 CaoController::AddExtension メソッドのオプション文字列

（共通）

設定項目	設定内容	必須	備考
APIType	<API type>	○	APIType を指定します． 以下の値が使用できます． 0：CSV データ登録 API 1：JSON データ登録 API 2：データレイク API（NoSQL） 3：データレイク API（RDB） 100：COLMINA エッジ IoT-PF API
APIVersion	<Version>	-	2018 年 3 月現在は"v1"を指定． 省略時は，"v1"が設定されます．

表 2-4 CaoController::AddExtension メソッドのオプション文字列

（CSV データ登録 API）

設定項目	設定内容	必須	備考
TenantID	<Tenant ID>	○	リソース所有テナントの識別子． COLMINA のご契約後の通知内容に従ってください．
UserName	<User name>	○	COLMINA V2 基本利用のためのユーザー名．
Password	<Password>	○	COLMINA V2 基本利用のためのパスワード．

表 2-5 CaoController::AddExtension メソッドのオプション文字列

（JSON データ登録 API・COLMINA エッジ IoT-PF API）

設定項目	設定内容	必須	備考
Port	<Port number>	-	ポート番号を指定. 省略時は, [APIType=1 時] 31443 [APIType=100 時] 8080 が設定されます.
AccessCode	<Access code>	○	アクセスコードは COLMINA のサービスポータルにて設定した値

表 2-6 CaoController::AddExtension メソッドのオプション文字列

（データレイク API（NoSQL）・データレイク API（RDB））

設定項目	設定内容	必須	備考
TenantID	<Tenant ID>	○	リソース所有テナントの識別子. COLMINA のご契約後の通知内容に従ってください.
UserID	<User ID>	○	ユーザーID.
Password	<Password>	○	パスワード.
APIKey	<API key>	○	COLMINA V2 基本契約時に発行された API キー.

2.3.3. CaoExtension::AddVariable メソッド

CaoExtension の Execute コマンドの一部を CaoVariable オブジェクトの get/put で実行するための CaoVariable オブジェクトを追加します.

書式　AddVariable (<bstrName:BSTR>, [<bstrOption:BSTR>])

< bstrName > : [in] 変数名
以下のシステム変数名, 或いはユーザー変数名を指定できます.
システム変数 : '@'で始まる以下の予約文字列を指定します.
[APIType=0 時]

@ID
@RegistCSV
[APIType=1 時]
@ID
@RegistJSON
@RegistCSV
@RegistTXT
@RegistBIN
[APIType=2，3 時]
@ID
@RegistJSON
@ReferenceLatestData
@RetrieveDataCount
[APIType=100 時]
@ID
@RegistJSON
@RegistCSV
@RegistTXT

ユーザー変数 : ‘@’で始まらない任意の文字列を指定します.この場合は，オプション文字列の Type を指定することが必須です.

<bstrOption> : [in] 対応する Execute コマンドごとに指定されたオプションをコンマ区切りで指定します.
ユーザー変数を指定した場合は，以下の Type オプションを指定してください. この Type オプションにより，対応する Execute コマンドが確定されます.
システム変数を指定した場合は，システム変数名により対応する Execute コマンドが確定されます.

Type : ユーザー変数の場合，振る舞いを決めるためにシステム変数名を指定します.
[APIType=0 時]
@ID
@RegistCSV
[APIType=1 時]
@ID
@RegistJSON

@RegistCSV
@RegistTXT
@RegistBIN
[APIType=2，3 時]
@ID
@RegistJSON
@ReferenceLatestData
@RetrieveDataCount
[APIType=100 時]
@ID
@RegistJSON
@RegistCSV
@RegistTXT
このオプションはシステム変数に対して は効果がありません.
例）
Type=@RegistJSON

対応する Execute コマンドごとに必要なオプションは次項に記載します.

2.3.3.1. @ID

システム変数名に"@ID "を指定して使用します. get・put することで，親エクステンションの ID プロパティを取得・設定することが可能です.

2.3.3.2. @RegistJSON

[APIType=1，2，3，100 時]

システム変数名，もしくはユーザー変数の Type オプションに"@RegistJSON"を指定して使用します. JSON 文字列（BSTR）を put することで，"RegistJSON"コマンドと同様にリソースに対して JSON データを登録することが可能です.

この変数を CaoExtension に追加するためには，ユーザー変数指定時の Type オプション以外に，以下のオプションを指定してください.

表 2-7 @RegistJSON のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパスを指定します.
Date	-	[APIType=1，100 時]

		登録データに付与する登録日時(*1)を指定します. 省略時はリクエスト受信日時を採用します.
Retain	-	[APIType=1 時] MQTT broker 側で本登録データを保持しておくかどうかを指定します. ・true :保持する ・false :保持しない 省略時は false が指定されたものとします. ※Bulk Insert 指定時は, RETAIN が指定されても無視します.
BulkInsert	-	[APIType=1, 2, 3 時] Bulk Insert(1度に複数リクエストを送信すること.)を実行するか否かを指定します. [APIType=1 時] ・none:Bulk Insert しません ・single_resource_path:単一リソースに対して Bulk Insert を実行します 省略時は none が指定されたものとします. [APIType=2, 3 時] ・true:します ・false:しません 省略時は false が指定されたものとします.

(*1)ISO8601(基本表記としてのミリ秒表現を使用)に従います(20141225T103612.001Z など). 精度はミリ秒(ミリ秒を省略した場合, 0 ミリ秒とみなします)です. 以降の「登録日時」はすべて同一仕様です. * 秒とミリ秒の区切りは「.」, タイムゾーン指定は「±hhmm」形式で省略時は「Z」を添えます. 本サービスがレスポンスに格納する場合, UTC を用います. 以降の日時指定はこの規則に従います.

2.3.3.3. @RegistCSV

[APIType=0, 1, 100 時]

システム変数名, もしくはユーザー変数の Type オプションに"@RegistCSV "を指定して使用します. CSV 文字列(BSTR)を put することで, " RegistCSV "コマンドと同様にリソースに対して CSV データを登録することが可能です.

この変数を CaoExtension に追加するためには, ユーザー変数指定時の Type オプション以外に, 以下のオプションを指定してください.

表 2-8 @RegistCSV のオプション文字列

設定項目	必須	備考
TableID	○	[APIType=0 時] 構造化データを登録する RDB テーブルの識別子.
ResourcePath	○	[APIType=1, 100 時] リソースパスを指定します.
Date	-	[APIType=0 時] 送信データに付与する送信日時(*1)を指定します. 省略時は POST 日時を採用します. [APIType=1, 100 時] 登録データに付与する登録日時を指定します. 省略時はリクエスト受信日時を採用します.
FormatCheck	-	[APIType=0 の時] フォーマットチェックを実施するか否かを指定します. ・true :実施する ・false :実施しない 省略時は true が指定されたものとします.
DataHeader	-	[APIType=0 の時] データのヘッダが存在するか否かを指定します. ・true :存在する ・false :存在しない 省略時は false が指定されたものとします.
Option1	-	[APIType=0 の時] 任意項目 1(必要に応じて設定)を指定します. 省略時は何もセットしません.
Option2	-	[APIType=0 の時] 任意項目 2(必要に応じて設定)を指定します. 省略時は何もセットしません.
Option3	-	[APIType=0 の時] 任意項目 3(必要に応じて設定)を指定します. 省略時は何もセットしません.
Retain	-	[APIType=1 時] MQTT broker 側で本登録データを保持するか否かを指定します. ・true :保持する ・false :保持しない

		省略時は false が指定されたものとします.
Skip	-	[APIType=1, 100 時] Body データの先頭から削除する行数を指定します. 省略時は行削除を行いません.
NumConv	-	[APIType=1, 100 時] Body データ中の数値を文字列に変換するか否かを指定します. ・true :数値変換する. ・false :数値変換しない. 省略時は false が指定されたものとします.

(*1)YYYY-MM-DDThh:mm:ssTz の形式に従います.(例:2016-12-25T10:36:12+09:00). 精度はミリ秒(ミリ秒を省略した場合, 0 ミリ秒とみなします).

2.3.3.4. @RegistTXT

[APIType=1, 100 時]

システム変数名, もしくはユーザー変数の Type オプションに"@RegistTXT "を指定して使用します. 文字列(BSTR)を put することで, " RegistTXT "コマンドと同様にリソースに対して TXT データを登録することが可能です.

この変数を CaoExtension に追加するためには, ユーザー変数指定時の Type オプション以外に, 以下のオプションを指定してください.

表 2-9 @RegistTXT のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパスを指定します.
Date	-	登録データに付与する登録日時を指定します. 省略時はリクエスト受信日時を採用します.
Retain	-	[APIType=1 時] MQTT broker 側で本登録データを保持するか否かを指定します. ・true :保持する ・false :保持しない 省略時は false が指定されたものとします.

2.3.3.5. @RegistBIN

[APIType=1 時]

システム変数名, もしくはユーザー変数指定時の Type オプションに"@RegistBIN"を指定して使用します. バイナリ(ARRAY|UI1)を put することで"RegistBIN"コマンドと同様にリソースに対して新たにバイナリデータ

を登録することが可能です.

この変数をCaoExtensionに追加するためには，ユーザー変数指定時のTypeオプション以外に，以下のオプションを指定してください.

表 2-10 @RegistBIN のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパスを指定します.
MimeType	○	Content-Type に指定する MIME-TYPE を指定します.
Date	-	登録データに付与する登録日時を指定します. 省略時はリクエスト受信日時を採用します.
Retain	-	MQTT broker 側で本登録データを保持するか否かを指定します. ・true :保持する ・false :保持しない 省略時は false が指定されたものとします.
Compression	-	Body データを圧縮して送信する際の圧縮タイプを以下で指定します. ・gz 省略時は Body データが無圧縮とみなします.

2.3.3.6. @ReferenceLatestData

[APIType=2, 3 時]

システム変数名，もしくはユーザー変数の Type オプションに"@ReferenceLatestData"を指定して使用します. get することで"ReferenceLatestData"コマンドと同様に，非構造化データ・構造化データを文字列（BSTR）として取得することが可能です.

この変数をCaoExtensionに追加するためには，ユーザー変数指定時のTypeオプション以外に，以下のオプションを指定してください.

表 2-11 @ReferenceLatestData のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパスを指定します.
Accept	-	[APIType=3 時] データ形式を指定します. ・application/json：JSON 形式 ・text/csv：CSV 形式

		省略時は application/json が指定されたものとして扱います.
Doublequote	-	[APIType=3 時] 数値項目のダブルクォートの有無を指定します. ・true:数値項目を文字列(ダブルクォートあり)の形式で返却 ・false:数値項目をそのまま(ダブルクォートなし)で返却 省略時は true が指定されたものとして扱います.

2.3.3.7. @RetrieveDataCount

[APIType=2, 3 時]

システム変数名, もしくはユーザー変数の Type オプションに"@RetrieveDataCount"を指定して使用します. get することで"RetrieveDataCount"コマンドと同様に, 非構造化データ・構造化データの HIT データ数(I4)を取得することが可能です.

この変数を CaoController に追加するためには, ユーザー変数指定時の Type オプション以外に, 以下のオプション指定をしてください.

表 2-12 @RetrieveDataCount のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパスを指定します.
Filter	-	QUERY の$filter パラメータを指定します. <検索条件>に一致するもののみを返すよう, 結果を限定します. <検索条件>は,「要素 演算子 値」とし, and or で複数定義可能です. 使用可能な演算子, 要素は後述します. このオプションの詳細は 5.2.1.を参照してください.

2.4. 変数一覧

COLMINA プロバイダでは以下のシステム変数が予約されています. またユーザー変数には任意の名前を使用することができます.

変数名	データ型	説明	属性	
			get	put
@RegistJSON	VT_BSTR	[APIType=1, 2, 3, 100 時] リソース_JSON・非構造化データ・構造化データへの JSON 文字列のデータ登録(蓄積)を行います.	-	○
@RegistCSV	VT_BSTR	[APIType=0, 1, 100 時] リソース_JSON への CSV 文字列のデータ登録(蓄積)を行います.	-	○

@RegistTXT	VT_BSTR	[APIType=1, 100 時] リソース_JSON への TXT 文字列のデータ登録(蓄積)を行います.	-	○
@RegistBIN	VT_ARRAY| VT_UI1	[APIType=1 時] リソース_JSON へのバイナリデータ登録(蓄積)を行います.	-	○
@ReferenceLatestData	VT_BSTR	[APIType=2, 3 時] 非構造化データ・構造化データの最新データ参照を行います.	○	-
@RetrieveDataCount	VT_I4	[APIType=2, 3 時] 非構造化データ・構造化データの HIT データ数を取得します.	○	-

3. コマンドリファレンス

3.1. Extension クラス

表 3-1 CaoExtension::Execute コマンド一覧

種別	コマンド	機能	対応変数	ページ
汎用 REST	REST	[APIType=1, 100 時] ユーザー指定された URL パス, クエリを付与して PUT, GET, DELETE, POST を実行します.	-	21
データ登録/転送	RegistJSON	[APIType=1, 2, 3, 100 時] リソース_JSON・非構造化データ・構造化データへの JSON 文字列のデータ登録(蓄積)を行います.	@RegistJSON	23
	RegistCSV	[APIType=0, 1, 100 時] リソース_JSON への CSV 文字列のデータ登録(蓄積)を行います.	@RegistCSV	25
	RegistTXT	[APIType=1, 100 時] リソース_JSON への TXT 文字列のデータ登録(蓄積)を行います.	@RegistTXT	28
	RegistBIN	[APIType=1 時] リソース_JSON へのバイナリデータ登録(蓄積)を行います.	@RegistBIN	29
非構造化データ・構造化データの参照	ReferenceLatestData	[APIType=2, 3 時] 非構造化データ・構造化データの最新データ参照を行います.	-	30
	ReferencePastData	[APIType=2, 3 時] 非構造化データ・構造化データの過去データ(指定日時)参照を行います	-	32
非構造化データ・構造化データの検索	RetrieveData	[APIType=2, 3 時] 非構造化データ・構造化データのデータ検索を行います.	-	33

	RetrieveDataCount	[APIType=2, 3 時] 非構造化データ・構造化データのHITデータ数を取得します.	-	36
非構造化データ・構造化データの更新	UpdateData	[APIType=2, 3 時] 非構造化データ・構造化データのデータの更新を行います.	-	36
非構造化データ・構造化データの削除	DeleteData	[APIType=2, 3 時] 非構造化データ・構造化データのデータの削除を行います.	-	37

3.1.1. 汎用 REST

3.1.1.1. CaoExtension::Execute("REST") コマンド

[APIType=1, 100 時]

富士通 COLMINA に対して, PUT, GET, DELETE, POST を汎用的に実行します. レスポンスの Content は UTF-8 の文字列として解釈されます.

書式　REST (<Data>)

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <REST メソッド>(BSTR)
必須要素
以下のいずれかを指定します.
・PUT
・GET
・DELETE
・POST

第 2 要素 = <基本となる URL に付加するパス>(BSTR)
必須要素
AddController と AddExtension 時に指定したパラメータと本要素で指定されたパスからリクエスト先の URL を作成します.
作成される URL は以下の通りです.
＜Base URI＞/＜API バージョン＞/＜テナント ID＞/＜基本となる URL に付加するパス＞

第 3 要素 = <URL パスに付加するクエリ文字列>(BSTR)
省略可能要素(*1)
先頭に「?」を付けずに指定してください.
例）"KeyA=ValueA&KeyB=ValueB"

第 4 要素 = <Content-Type>(BSTR)
省略可能要素(*1)
省略時は Content-Type は指定されません.

第 5 要素 = <送信する Body データ>(ARRAY|UI1)
省略可能要素(*1)
バイナリで送信する Content を設定します.

戻り値　　:　[out] (ARRAY|VARIANT)

第 1 要素　　=　StatusCode（int）
第 2 要素　　=　Header（BSTR）
第 3 要素　　=　レスポンスの Body 文字列（BSTR）

(*1)値に null か空文字を指定することにより，その要素を省略したとみなされます．また，ある要素以降をすべて省略する場合は，指定する要素のみの配列（すべての要素が 5 つある中で先頭から 3 要素のみ指定する場合は，長さが 3 の配列）を引数に渡すこともできます．要素を省略した結果，一要素の配列となった場合は，配列型ではなく中身の要素のみを引数として渡すこともできます．これは以降すべての Execute コマンドの引数に配列をとる場合に適用されます．

3.1.2. リソース_JSON・CSV データ・非構造化データ・構造化データへのデータ登録/転送

3.1.2.1. CaoExtension::Execute("RegistJSON") コマンド

[APIType=1, 2, 3, 100 時]

リソースに対して新たに JSON データを登録/転送します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RegistJSON (<Data>)

[APIType=1, 100 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素

第 2 要素 = <送信する JSON 文字列>(BSTR)
必須要素

第 3 要素 = <登録データに付与する登録日時>(BSTR)
省略可能要素
省略時はリクエスト受信日時を採用します.

第 4 要素 = [APIType=1 時]
<RETAIN>(BOOL)
省略可能要素
MQTT broker 側で本登録データを保持するか否かを指定します.
・true :保持する
・false :保持しない
省略時は false が指定されたものとします.
※Bulk Insert 指定時は, RETAIN が指定されても無視します.

第 5 要素 = [APIType=1 時]
<Bulk Insert フラグ>(BSTR)
省略可能要素
Bulk Insert(1度に複数リクエストを送信すること.)を実行するか否かを指定します.
・none:Bulk Insert しません
・single_resource_path:単一リソースに対して Bulk Insert を実行します

省略時は none が指定されたものとします.

[APIType=2，3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>（BSTR）
必須要素
[APIType=2 時]
{データベース名}/{コレクション名}
[APIType=3 時]
{データベース名}/{スキーマ名}/{テーブル ID}

第 2 要素 = <送信する JSON 文字列>（BSTR）
必須要素

第 3 要素 = <Bulk Insert フラグ>（BOOL）
省略可能要素
Bulk Insert（1度に複数リクエストを送信すること.）を
実行するか否かを指定します.
・false：Bulk Insert を実行しません
・true：Bulk Insert を実行します
省略時は false が指定されたものとします.

3.1.2.2. CaoExtension::Execute("RegistCSV") コマンド

[APIType=0, 1, 100 時]

リソースに対して新たに CSV データを登録/転送します

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式 RegistCSV (<Data>)

[APIType=0 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <テーブル ID>(BSTR)
必須要素

第 2 要素 = <送信する CSV 文字列>(BSTR)
必須要素

第 3 要素 = <登録データに付与する送信日時>(BSTR)
省略可能要素
省略時は POST 日時を採用します.

第 4 要素 = <フォーマットチェック実施>(BOOL)
省略可能要素
フォーマットチェックを実施するか否かを指定します.
・true :実施する
・false :実施しない
省略時は true が指定されたものとします.

第 5 要素 = <データヘッダ有無>(BOOL)
省略可能要素
データのヘッダが存在するか否かを指定します.
・true :存在する
・false :存在しない
省略時は false が指定されたものとします.

第 6 要素 = <オプション1>(BSTR)
省略可能要素
任意項目 1(必要に応じて設定)を指定します.
省略時は何もセットしません.

第 7 要素 = <オプション 2>(BSTR)
省略可能要素
任意項目 2(必要に応じて設定)を指定します.

省略時は何もセットしません.

第 8 要素 = <オプション 3>(BSTR)
省略可能要素
任意項目 3(必要に応じて設定)を指定します.
省略時は何もセットしません.

[APIType=1 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
第 2 要素 = <送信する CSV 文字列>(BSTR)
必須要素
第 3 要素 = <登録データに付与する登録日時>(BSTR)
省略可能要素
省略時はリクエスト受信日時を採用します.
第 4 要素 = <RETAIN>(BOOL)
省略可能要素
MQTT broker 側で本登録データを保持するか否かを指定します.
・true :保持する
・false :保持しない
省略時は false が指定されたものとします.
第 5 要素 = <Body データ削除指定行>(I4)
省略可能要素
Body データの先頭から削除する行数を指定します.
省略時は行削除を行いません.
第 6 要素 = <数値変換>(BOOL)
省略可能要素
Body データ中の数値を文字列に変換するか否かを指定します.
・true :数値変換する.
・false :数値変換しない.
省略時は true が指定されたものとします.

[APIType=100 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
第 2 要素 = <送信する CSV 文字列>(BSTR)
必須要素
第 3 要素 = <登録データに付与する登録日時>(BSTR)
省略可能要素
省略時はリクエスト受信日時を採用します.
第 4 要素 = <Body データ削除指定行>(I4)
省略可能要素
Body データの先頭から削除する行数を指定します.
省略時は行削除を行いません.
第 5 要素 = <数値変換>(BOOL)
省略可能要素
Body データ中の数値を文字列に変換するか否かを指定します.
・true :数値変換する.
・false :数値変換しない.
省略時は true が指定されたものとします.

3.1.2.3. CaoExtension::Execute("RegistTXT") コマンド

[APIType=1, 100 時]

リソースに対して新たに TXT データを登録/転送します

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RegistTXT (<Data>)

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素

第 2 要素 = <送信する TXT 文字列>(BSTR)
必須要素

第 3 要素 = <登録データに付与する登録日時>(BSTR)
省略可能要素
省略時はリクエスト受信日時を採用します.

第 4 要素 = [APIType=1 時]
<RETAIN>(BOOL)
省略可能要素
MQTT broker 側で本登録データを保持するか否かを指定します.
・true :保持する
・false :保持しない
省略時は false が指定されたものとします.

3.1.2.4. CaoExtension::Execute("RegistBIN") コマンド

[APIType=1 時]

リソースに対して新たにバイナリデータを登録/転送します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RegistBIN (<Data>)

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素

第 2 要素 = <Content-Type に指定する MIME-TYPE>(BSTR)
必須要素

第 3 要素 = <送信するバイナリ>(ARRAY|UI1)
必須要素

第 4 要素 = <登録データに付与する登録日時>(BSTR)
省略可能要素
省略時はリクエスト受信日時を採用します.

第 5 要素 = <RETAIN>(BOOL)
省略可能要素
MQTT broker 側で本登録データを保持するか否かを指定します.
・true :保持する
・false :保持しない
省略時は false が指定されたものとします.

第 6 要素 = <圧縮タイプ>(BSTR)
省略可能要素
Body データを圧縮して送信する際の圧縮タイプを以下で指定します.
・gz
省略時は Body データが無圧縮とみなします.

3.1.3. 非構造化データ・構造化データの参照

3.1.3.1. CaoExtension::Execute("ReferenceLatestData")

[APIType=2, 3 時]

非構造化データ・構造化データに登録済みの最新データを参照します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　ReferenceLatestData (<Data>)

[APIType=2 時]

<Data> : [in] BSTR
<リソースパス>
必須要素
{データベース名}/{コレクション名}

戻り値 : [out] (BSTR)
参照したデータ.

[APIType=3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
{データベース名}/{スキーマ名}/{テーブル ID}

第 2 要素 = <データ形式>(BSTR)
省略可能要素
・application/json:JSON 形式
・text/csv:CSV 形式
省略時は application/json が指定されたものとして扱います.

第 3 要素 = <数値項目のダブルクォートの有無>(BOOL)
省略可能要素
・true:数値項目を文字列(ダブルクォートあり)の形式で返却
・false:数値項目をそのまま(ダブルクォートなし)で返却
省略時は true が指定されたものとして扱います.

戻り値　　:　[out] (BSTR)
参照したデータ.

3.1.3.2. CaoExtension::Execute("ReferencePastData")

[APIType=2, 3 時]

登録日時が指定日時と一致する, 非構造化データ・構造化データに登録済みのデータを参照します.
不正な引数を指定した場合は E_INVALIDARG が発生します.
(*1) <登録日時>のデータが複数存在した場合, 全てのデータが返却されます

書式　ReferencePastData (<Data>)

[APIType=2 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
{データベース名}/{コレクション名}
第 2 要素 = <登録日時>(BSTR)
必須要素
参照対象データの登録日時

戻り値 : [out] (BSTR)
参照したデータ.

[APIType=3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
{データベース名}/{スキーマ名}/{テーブル ID}
第 2 要素 = <登録日時>(BSTR)
必須要素
参照対象データの登録日時
第 3 要素 = <データ形式>(BSTR)
省略可能要素
・application/json:JSON 形式
・text/csv:CSV 形式
省略時は application/json が指定されたものとして扱います.

第 4 要素 = <数値項目のダブルクォートの有無>(BOOL)
省略可能要素
・true:数値項目を文字列(ダブルクォートあり)の形式で返却
・false:数値項目をそのまま(ダブルクォートなし)で返却
省略時は true が指定されたものとして扱います.

戻り値 : [out] (BSTR)
参照したデータ.

3.1.4. 非構造化データ・構造化データの検索

3.1.4.1. CaoExtension::Execute("RetrieveData")

[APIType=2, 3 時]

指定条件に一致する非構造化データ・構造化データに登録済みのデータを検索します.
不正な引数を指定した場合は E_INVALIDARG が発生します.
(*1) [APIType=2, 3 時]検索結果として得られるデータの順番は, <登録日時>でソートします

書式 RetrieveData (<Data>)

[APIType=2 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
{データベース名}/{コレクション名}
第 2 要素 = <検索条件>(BSTR)
省略可能要素
<検索条件>に一致するもののみを返すよう, 結果を限定します.
詳細は 5.2.1.を参照してください.
第 3 要素 = <取得件数>(I4)
省略可能要素
検索結果として得られるデータを n 件に限定します.
最大取得件数は 20,000 件です.
省略時は上限なしとして扱います.
第 4 要素 = <スキップ件数>(I4)

省略可能要素
検索結果として得られたデータを n 件 skip します. 検索結果として得られるデータの順番は, <登録日時>でソートします.
省略時は 0 が指定されたものとして扱います.

戻り値 : [out] (BSTR)
一致した非構造化データ.

[APIType=3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
{データベース名}/{スキーマ名}/{テーブル ID}

第 2 要素 = <データ形式>(BSTR)
省略可能要素
・application/json:JSON 形式
・text/csv:CSV 形式
省略時は application/json が指定されたものとして扱います.

第 3 要素 = <数値項目のダブルクォートの有無>(BOOL)
省略可能要素
・true:数値項目を文字列(ダブルクォートあり)の形式で返却
・false:数値項目をそのまま(ダブルクォートなし)で返却
省略時は true が指定されたものとして扱います.

第 4 要素 = <検索条件>(BSTR)
省略可能要素
<検索条件>に一致するもののみを返すよう, 結果を限定します.
詳細は 5.2.1.を参照してください.

第 5 要素 = <取得件数>(I4)
省略可能要素
検索結果として得られるデータを n 件に限定します.
最大取得件数は 20,000 件です.
省略時は上限なしとして扱います.

第 6 要素 = <スキップ件数>(I4)
省略可能要素
検索結果として得られたデータを n 件 skip します. 検索結果として得られるデータの順番は, <登録日時>でソートします.
省略時は 0 が指定されたものとして扱います.

戻り値 : [out] (BSTR)
一致した構造化データ.

3.1.4.2. CaoExtension::Execute("RetrieveDataCount")

[APIType=2, 3]

指定条件に一致する非構造化データ・構造化データに登録済みのデータ件数を取得します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RetrieveDataCount (<Data>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<リソースパス>(BSTR)
必須要素
[APIType=2 時]
{データベース名}/{コレクション名}
[APIType=3 時]
{データベース名}/{スキーマ名}/{テーブル ID}

第 2 要素　=　<検索条件>(BSTR)
省略可能要素
<検索条件>に一致するもののみを返すよう, 結果を限定します.
詳細は 5.2.1.を参照してください.

戻り値　:　[out] (I4)
一致した非構造化データ・構造化データのデータ数.

3.1.5. 非構造化データ・構造化データの更新

3.1.5.1. CaoExtension::Execute("UpdateData")

[APIType=2 時]

指定したオブジェクト ID に一致する非構造化データに登録済みのデータを指定したデータで更新します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

[APIType=3 時]

指定したデータ中のプライマリキーに一致する構造化データに登録済みのデータを指定したデータで更新します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　UpdateData (<Data>)

[APIType=2, 3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
[APIType=2 時]
{データベース名}/{コレクション名}/{オブジェクト ID}
[APIType=3 時]
{データベース名}/{スキーマ名}/{テーブル ID}
第 2 要素 = <更新するデータ>(BSTR)
必須要素

3.1.6. 非構造化データ・構造化データの削除

3.1.6.1. CaoExtension::Execute("DeleteData")

[APIType=2, 3 時]

指定条件に一致する非構造化データ・構造化データに登録済みのデータを削除します.
不正な引数を指定した場合は E_INVALIDARG が発生します.

書式 DeleteData (<Data>)

[APIType=2, 3 時]

<Data> : [in] (ARRAY|VARIANT)

第 1 要素 = <リソースパス>(BSTR)
必須要素
[APIType=2 時]
{データベース名}/{コレクション名}
[APIType=3 時]
{データベース名}/{スキーマ名}/{テーブル ID}
第 2 要素 = <検索条件>(BSTR)
必須要素
<検索条件>に一致するものを削除します.
詳細は 5.2.1.を参照してください.

4. エラーコード

富士通 COLMINA プロバイダでは，以下の固有エラーコードが定義されています．

ORiN2 共通エラーコードについては，「ORiN2 プログラミングガイド」のエラーコードの章をご参照ください．

表 4-1 固有エラーコード

エラー名	エラー番号	説明
RemoteCertificateNotAvailable	0x80100000	証明書が利用できない場合に返されます．
RemoteCertificateNameMismatch	0x80100001	証明書名が不一致の場合に返されます．
RemoteCertificateChainErrors	0x80100002	ChainStatus が，空でない配列を返した場合に返されます．

5. 補足

5.1. くリ ソースパス(/$all 利用可)>の記述方法

- リソースパスをフルパスで指定
  指定したリソースパスのリソースデータを返却します.

- リソースパスの途中までを指定し, その後ろに「/$all」を付加
  指定したパス配下全てのリソースパスのリソースデータを返却します.
  例)
  「AX」「A/B」「A/B/C」の3つのリソースが存在する状態で,「A/$all」を指定した場合,「A/B」および「A/B/C」の2つのリソースを対象とします.

5.2. 検索条件について

5.2.1. データレイク API(NoSQL・RDB)

5.2.1.1. 検索条件の演算子

Filter オプションで指定できる検索条件には以下の演算子を利用できます.

項番	演算子	説明	例
1	eq	等号	Filter=owner eq 'Tom'
2	ne	不等号	Filter=owner ne null
3	gt	より大きい	Filter=value gt 100
4	ge	以上	Filter=value ge 100
5	lt	より小さい	Filter=value lt 100
6	le	以下	Filter=value le 100
7	and	論理積	Filter=value ge 100 and owner eq 'Tom'
8	or	論理和	Filter=value ge 100 or owner eq 'Tom'

上記以外の演算子は使用できません.

※ただし, 優先順位を表す「(」, 「)」については使用できます

5.2.1.2. 検索条件に利用可能な要素

Filter オプションの比較式の左辺には以下の要素が利用できます.

項番	要素名	説明	備考
1	_date	登録日時	検索対象レコードの登録日時, ISO8601 形式「YYYYMMDDThhmmss.SSSZ」に従います 引用符「'」で囲みません
2	任意の項目名	登録データの項目	※NoSQL のみ key が階層構造になっている場合は, "."(ピリオド)を区切り文字に使用して「Floor.Value」のように指定します. 深さは最大 15
3	_id	ドキュメントのオブジェクト ID ※NoSQL のみ	比較式の右辺は引用符「'」で囲みます
4	_resource_path	リソースパス文字列 ※NoSQL のみ	比較式の右辺は引用符「'」で囲みます

Filter オプションの比較式の右辺には以下の要素が利用できます.

項番	要素名	説明	備考
1	_date	登録日時	検索対象レコードの登録日時
2	任意の項目名	登録データの項目	---

3	null	NULL 値	NULL 値に対しては等号・不等号以外の演算子は使用できません
4	'任意の値'	文字列	引用符「'」で囲みます 引用符そのものを記述する場合は二つ重ねて「''」を設定します
5	任意の日時	日時リテラル	引用符「'」で囲みません タイムゾーン付項目の場合は ISO8601 基本形式「YYYYMMDDThhmmss.SSSZ」に従います タイムゾーンなし形式の場合は YYYY-MM-DD hh:mm:ss[.SSS]形式に従います
6	任意の数値	数値リテラル	引用符「'」で囲みません

5.2.1.3. あいまい検索

Filter オプションを利用し，前方一致/後方一致/部分一致のあいまい条件で検索をすることができます．
あいまい検索時は，Filter オプションの値フィールドにワイルドカード(*)を指定します．

- 利用可能な演算子
  "eq"，"ne"の 2 つの演算子が利用できます．
  ※"eq"，"ne"を用いたあいまい検索の論理式を"and"，"or"で結合した複合条件の検索もできます

- 利用可能な要素/値
  Filter オプションの値フィールドに文字列（シングルクォートで囲む）を指定する場合のみワイルドカードを指定ができます．ワイルドカード（*）は，値フィールドの先頭もしくは末尾のみに指定ができます．
  ※先頭もしくは末尾の"*"はワイルドカードとみなし，途中の"*"は文字列とみなします．先頭もしくは末尾に文字としての"*"指定はできません

  値フィールドに"*"のみが指定された場合，以下の動作とします．
  ・"*"，"**" ： すべてのデータにマッチ
  ・"***" ： 文字列中に「*」を含むデータにマッチ

  注意事項
  データレイク API（NoSQL）でシステムにより自動付与するフィールドのうち，"_id"，"_date"と要素名に指定するあいまい検索は未対応です．

前方一致/後方一致/部分一致のあいまい検索実行時間は，検索条件が同一の場合，部分一致/後方一致より前方一致のほうが，検索時間が短くなります．

5.3. REST 時の JSON フォーマット

本フォーマットは，下記 URL の API リファレンスを元に作成しております.

https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/apireference.pdf

5.4. REST 時のレスポンスステータスコード

コマンド実行時に返却されるステータスコードの一部を紹介します.

Status-Code	Reason-Phrase	説明
200	OK	成功, リソースデータ作成の成功
201	Created	リソース, アクセスコード, イベント作成の成功
204	No Content	以下の何れかに該当 ・リソースデータ参照時において, リソースは存在するが 該当するリソースデータが存在しない ・リソースの削除において, 削除成功 ・メタデータ・アクセスコード・イベント参照時において, 各種情報が存在しない (将来変更する場合があります.) ・メタデータ・アクセスコード・イベント削除時において, 削除成功
206	Partial content	部分取得の成功
400	Bad Request	リクエストデータに不正値があります
401	Unauthorized	リソースへのアクセス権がありません
403	Forbidden	アクセス権がありません
404	Not Found	リソースが存在しません
405	Method Not Allowed	該当のメソッドタイプは許可されていません
408	Request Time-out	リクエストタイムアウトです
409	Conflict	他のリソースと競合しています
411	Length Required	サーバアクセスを拒否しました(Content-Lengh 指定なし)
412	Precondition Failed	サーバアクセスを拒否しました(リクエスト条件が不正)
413	Payload Too Large	サーバアクセスを拒否しました(リクエストボディサイズがサーバ許容範囲超越)
414	URI Too Long	サーバアクセスを拒否しました(URI が長い)
415	Unsupported Media Type	サーバアクセスを拒否しました(未サポート Content-Type)
416	Requested Range Not Satisfiable	サーバアクセスを拒否しました(Range 要求の値が不正)
421	Misdirected Request	レスポンスを生成できないサーバに送信されました
423	Locked	リソースがロックされています
429	Too Many Request	契約上のトラフィック上限を超えています

495	SSL Certificate Error	無効なクライアント証明書を受信しました
496	SSL Certificate Required	クライアントからクライアント証明書が送付されませんでした
497	HTTP Request Sent to HTTPS Port	HTTPS リクエストポートにて HTTP リクエストを受信しました
500	Internal Server Error	サーバ側の問題による失敗です
501	Not Implemented	サーバで未サポートリクエストのメソッドが送信されました
502	Bad Gateway	ゲートウェイサーバが起動していません
503	Service Unavailable	一時的にアクセスできません
504	Gateway Time-out	ゲートウェイサーバが時間内にレスポンスを返せませんでした

より詳細な情報は，下記 URL をご参照下さい.

https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/apireference.pdf

6. サンプルプログラム

以下に C# のサンプルを示します.

6.1. Variable への値の設定と取得

変数一覧にて紹介された Variable の使用例を示します.

6.1.1. 値の設定

List 6-1-1

```
…(略)…
using ORiN2.ManagedCAO;

namespace COLMINA_Sample
{
    public  class Sample
    {
        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly int API_TYPE = 1;  // JSONデータ登録API
        private readonly string ACCESS_CODE = "AccessCode";
        private readonly string RESOURCE_OPT = "ResourcePath=Test";
        private readonly string FILE_PATH = "jpegFilePath";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = null;
        private CCaoExtension m_CaoExtension = null;

        private void executeAllVariable()
        {
            // CAOエンジン生成
            m_CaoEngine = new CCaoEngine();
            m_CaoWorkspaces = m_CaoEngine.Workspaces;
            m_CaoWorkspace = m_CaoWorkspaces[0];

            // コントローラ引数
            var opstionStr = string.Format("BaseURL={0}", BASE_URL);

            // コントローラに接続
            m_CaoController = m_CaoWorkspace.AddController("Test",
                "CaoProv.FUJITSU.COLMINA", null, opstionStr);
```

```
// エクステンション引数
var extOptionStr = string.Format("APIType={0},AccessCode={1}", API_TYPE,
    ACCESS_CODE);

// エクステンションに接続
m_CaoExtension = m_CaoController.AddExtension("Ext", extOpstionStr);

// オプション文字列
var registBinOpt = string.Format("{0}, MimeType=image/jpeg, Retain=true",
    RESOURCE_OPT);

// @RegistJSON
CCaoVariable systemVariable = m_CaoExtension.AddVariable("@RegistJSON",
    RESOURCE_OPT);
CCaoVariable userVariable = m_CaoExtension.AddVariable("RegistJSON",
    string.Format("Type=@RegistJSON,{0}", RESOURCE_OPT));

string jsonStr = "{¥"test¥":1}";
systemVariable.Value = jsonStr;
userVariable.Value = jsonStr;

// @RegistCSV
systemVariable = m_CaoExtension.AddVariable("@RegistCSV", RESOURCE_OPT);
userVariable = m_CaoExtension.AddVariable("RegistCSV",
    string.Format("Type=@RegistCSV, {0}", RESOURCE_OPT));
string csvStr = @"name,age,tel
    田中,30,012-345-6789
    鈴木,40,098-765-4321";

systemVariable.Value = csvStr;
userVariable.Value = csvStr;

// @RegistTXT
systemVariable = m_CaoExtension.AddVariable("@RegistTXT", RESOURCE_OPT);
userVariable = m_CaoExtension.AddVariable("RegistTXT",
    string.Format("Type=@RegistTXT,{0}", RESOURCE_OPT));
```

```
                string txtStr = "test";
                systemVariable.Value = txtStr;
                userVariable.Value = csvStr;

                systemVariable = m_CaoExtension.AddVariable("@RegistBIN", registBinOpt);
                userVariable = m_CaoExtension.AddVariable("RegistBIN",
                    string.Format("Type=@RegistBIN,{0}", registBinOpt));

                if (!string.IsNullOrEmpty(FILE_PATH))
                {
                    byte[] binaryArray = File.ReadAllBytes(FILE_PATH);
                    systemVariable.Value = binaryArray;
                    userVariable.Value = binaryArray;
                }
            }
        }
    }
```

6.1.2. 値の取得

List 6-2-2

```
…(略)…
using ORiN2.ManagedCAO;

namespace COLMINA_Sample
{
    public  class Sample
    {
        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly int API_TYPE = 3;  // データレイクAPI(NoSQL)
        private readonly string TENANT_ID = "TenantID";
        private readonly string USER_ID = "UserID";
        private readonly string PASSWORD = "Password";
        private readonly string API_KEY = "APIKey";
        private readonly string RESOURCE_OPT = "ResourcePath=Test";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = null;
        private CCaoExtension m_CaoExtension = null;

        private void executeAllVariable()
        {
            // CAOエンジン生成
            m_CaoEngine = new CCaoEngine();
            m_CaoWorkspaces = m_CaoEngine.Workspaces;
            m_CaoWorkspace = m_CaoWorkspaces[0];

            // コントローラ引数
            var opstionStr = string.Format("BaseURL={0}", BASE_URL);

            // コントローラに接続
            m_CaoController = m_CaoWorkspace.AddController("Test",
                "CaoProv.FUJITSU.COLMINA", null, opstionStr);

            // エクステンション引数
            var extOptionStr =
                string.Format("APIType={0},TenantID={1},UserID={2},Password={3},APIKey={
```

```
                4}", API_TYPE, TENANT_ID, USER_ID, PASSWORD, API_KEY);

            // エクステンションに接続
            m_CaoExtension = m_CaoController.AddExtension("Ext", extOpstionStr);

            // @ReferenceLatestData
            CCaoVariable systemVariable =
                m_CaoExtension.AddVariable("@ReferenceLatestData", RESOURCE_OPT);
            CCaoVariable userVariable = m_CaoExtension.AddVariable("ReferenceLatestData",
                string.Format("Type=@ReferenceLatestData,{0}", RESOURCE_OPT));

            var referenceLatestDataStr = systemVariable.Value.ToString();
            referenceLatestDataStr = userVariable.Value.ToString();
        }
    }
}
```

6.2. 汎用 REST

汎用 REST の使用例を示します.

List 6-2

```
…(略)…
using System.Collections.Generic;
using ORiN2.ManagedCAO;

namespace COLMINA_Sample
{
    public  class Sample
    {
        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly int API_TYPE = 1;  // JSONデータ登録API
        private readonly string ACCESS_CODE = "AccessCode";
        private readonly string RESOUCE_PATH = "Test";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = null;
        private CCaoExtension m_CaoExtension = null;

        private void executSample()
        {
            // CAOエンジン生成
            m_CaoEngine = new CCaoEngine();
            m_CaoWorkspaces = m_CaoEngine.Workspaces;
            m_CaoWorkspace = m_CaoWorkspaces[0];

            // コントローラ引数
            var opstionStr = string.Format("BaseURL={0}", BASE_URL);

            if (m_CaoController == null)
            {
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
                    "CaoProv.FUJITSU.COLMINA", null, opstionStr);
            }

            // エクステンション引数
            var extOptionStr = string.Format("APIType={0},AccessCode={1}", API_TYPE,
```

```
                    ACCESS_CODE);

                if (m_CaoExtension == null)
                {
                    // エクステンションに接続
                    m_CaoExtension = m_CaoController.AddExtension("Ext", extOpstionStr);
                }

                var paramList = new List<object>();

                // 汎用REST
                paramList.Add("PUT");
                paramList.Add(RESOUCE_PATH);
                paramList.Add(string.Empty);
                paramList.Add(string.Empty);
                paramList.Add(System.Text.Encoding.ASCII.GetBytes("{¥"abc¥":¥"ABC¥"}"));
                var restResult = m_CaoExtension.Execute("REST", paramList.ToArray());
            }
        }
    }
```

6.3. データ制御

6.3.1. リソース_JSON へのデータ登録/転送

リソース_JSON へのデータ登録/転送 の使用例を示します.

List 6-3-1

```
…(略)…
using System.IO;
using System.Collections.Generic;
using ORiN2.ManagedCAO;

namespace COLMINA_Sample
{
    public  class Sample
    {
        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly int API_TYPE = 1;  // JSONデータ登録API
        private readonly string ACCESS_CODE = "AccessCode";

        private readonly string RESOURCE_OPT = "ResourcePath=Test";
        private readonly string FILE_PATH = "jpegFilePath";
        private readonly string RESOUCE_PATH = "Test";
        private readonly string REGIST_DATE = "20180831T023219.222Z";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = null;
        private CCaoExtension m_CaoExtension = null;

        private void executSample()
        {
            // CAOエンジン生成
            m_CaoEngine = new CCaoEngine();
            m_CaoWorkspaces = m_CaoEngine.Workspaces;
            m_CaoWorkspace = m_CaoWorkspaces[0];

            // コントローラ引数
            var opstionStr = string.Format("BaseURL={0}", BASE_URL);

            if (m_CaoController == null)
            {
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
```

```
            "CaoProv.FUJITSU.COLMINA", null, opstionStr);
}

// エクステンション引数
var extOptionStr = string.Format("APIType={0},AccessCode={1}", API_TYPE,
    ACCESS_CODE);

if (m_CaoExtension == null)
{
    // エクステンションに接続
    m_CaoExtension = m_CaoController.AddController("Ext", extOpstionStr);
}

// オプション文字列
var registBinOpt = string.Format("{0}, MimeType=image/jpeg, Retain=true",
    RESOURCE_OPT);
var paramList = new List<object>();
var paramStrList = new List<string>();

// データ制御
// リソースJSONへのデータ登録/転送
// RegistJSON
paramList.Clear();
paramList.Add(RESOUCE_PATH);
paramList.Add("{¥"test¥":1}");
m_CaoExtension.Execute("RegistJSON", paramList.ToArray());

// RegistCSV
paramList.Clear();
string csvStr = @"name,age,tel
    田中,30,012-345-6789
    鈴木,40,098-765-4321";
paramList.Add(RESOUCE_PATH);
paramList.Add(csvStr);
m_CaoExtension.Execute("RegistCSV", paramList.ToArray());

// RegistTXT
```

```
                paramList.Clear();
                paramList.Add(RESOUCE_PATH);
                paramList.Add("testTxt");
                m_CaoExtension.Execute("RegistTXT", paramList.ToArray());

                // RegistBIN
                paramList.Clear();
                paramList.Add(RESOUCE_PATH);
                paramList.Add("MimeType=image/jpeg");
                if (!string.IsNullOrEmpty(FILE_PATH))
                {
                    paramList.Add(File.ReadAllBytes(FILE_PATH));
                    m_CaoExtension.Execute("RegistBIN", paramList.ToArray());
                }
            }
        }
    }
```

6.3.2. 非構造化データの参照

非構造化データの参照 の使用例を示します.

List 6-3-2

```
…(略)…
using System.IO;
using System.Collections.Generic;
using ORiN2.ManagedCAO;

namespace COLMINA_Sample
{
    public  class Sample
    {

        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly int API_TYPE = 3;  // データレイクAPI(NoSQL)
        private readonly string TENANT_ID = "TenantID";
        private readonly string USER_ID = "UserID";
        private readonly string PASSWORD = "Password";
        private readonly string API_KEY = "APIKey";
        private readonly string RESOURCE_OPT = "ResourcePath=Test";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = null;
        private CCaoExtension m_CaoExtension = null;

        private void executSample()
        {
            // CAOエンジン生成
            m_CaoEngine = new CCaoEngine();
            m_CaoWorkspaces = m_CaoEngine.Workspaces;
            m_CaoWorkspace = m_CaoWorkspaces[0];

            // コントローラ引数
            var opstionStr = string.Format("BaseURL={0}", BASE_URL);

            if (m_CaoController == null)
            {
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
```

```
                    "CaoProv.FUJITSU.COLMINA", null, opstionStr);
            }

            // エクステンション引数
            var extOptionStr = string.Format("APIType={0},AccessCode={1}", API_TYPE,
                ACCESS_CODE);

            if (m_CaoExtension == null)
            {
                // エクステンションに接続
                m_CaoExtension = m_CaoController.AddController("Ext", extOpstionStr);
            }

            var paramList = new List<object>();

            // ReferenceLatestData
            paramList.Add(RESOUCE_PATH);
            var referenceLatestData = m_CaoExtension.Execute("ReferenceLatestData",
                paramList.ToArray());
        }
    }
}
```