富士通 IoT Platform プロバイダ
REST API 対応

Version 1.0.0

ユーザーズガイド

August 31, 2018

【備考】

【改版履歴】

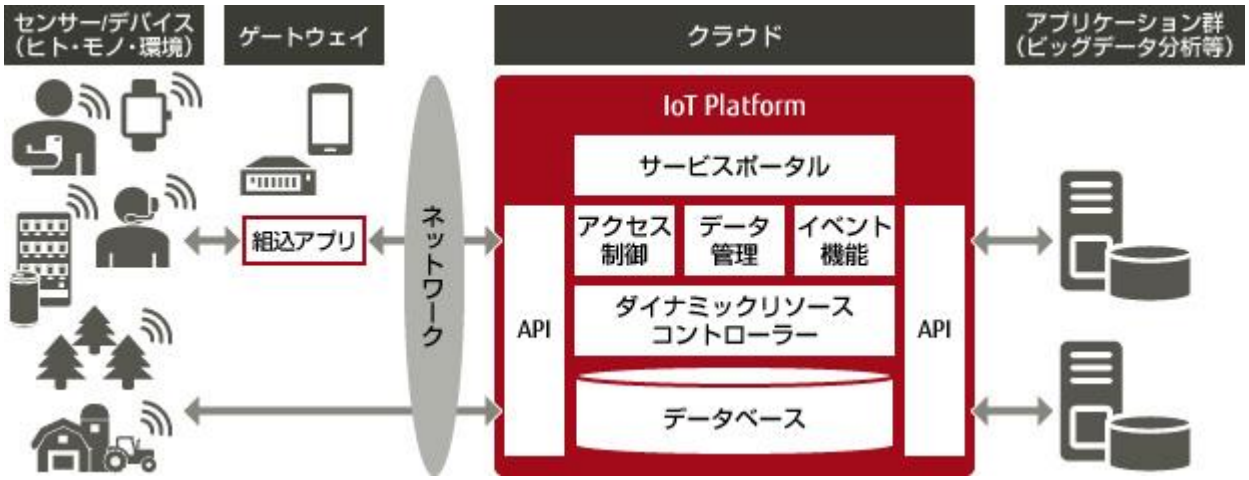
バージョン	日付	内容
1.0.0	2018-4-16	初版.
	2018-6-5	登録（JSON，CSV，TXT，BIN），最新データ参照，登録データ HIT 数取得コマンドを Variable として実装.
	2018-7-11	下記 API コマンドを Execute として実装. リソース制御 (登録，更新，削除) , アクセスコード制御 (登録，参照，更新，削除), イベント制御 (登録，参照，更新，削除)
	2018-7-20	REST 時のレスポンスステータスコード を追加
	2018-7-30	概要に API について簡易説明追記 filter 条件に例を追記, アクセスコード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)

目次

1. はじめに

IoT Platform プロバイダは，富士通 IoT Platformが提供する各種サービスを活用するための接続とAPI呼び出しを提供するプロバイダです．

このプロバイダを利用することで，IoT Platformコア上で製造業各社やサービス・プロバイダーが持つものづくりの各種サービスを展開し，ものづくりノウハウやIoT機器，PLC，ロボットなどの工場設備や機器の情報などに関する様々なナレッジを共有，活用することができるようになります．



本ドキュメントでは，IoT Platform 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

IoT Platform プロバイダモジュールは，下記の DLL 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 IoT Platform プロバイダ

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

※ 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

IoT Platform プロバイダは，富士通 IoT Platform 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， リソース制御， アクセスコード制御， イベント制御など)をラップした機能を提供します．

API の内容を簡単に説明すると， IoT Platform に対して
・汎用 REST にて， REST が実行可能です．
・リソース制御にて，データの登録先を作成可能です．
・データ制御にて，リソースに対してデータの登録などの操作可能です．
・アクセスコード制御にて，リソースに対するアクセス権の付与や制限が可能です．
・イベント制御にて，リソースに登録しているデータの値を条件に，
IoT Platform から HTTP 通知またはメール通知することが可能です．

API 関連情報について以下に詳細な使い方を記載していますので参照願います．
[API リファレンス]
https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/apireference.pdf
[ユーザガイド]
https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/userguide.pdf

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.IoTPlatform"
<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
- テナント ID
- アクセスコード
- API バージョン
- タイムアウト

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, APIVersion=v1,…"

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.
TenantID	<Tenant>	○	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.
AccessCode	<Access code>	○	アクセスコードは IoT Platform のサービスポータルにて設定した値
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.
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.

2.3.2. CaoController::AddVariable メソッド

CaoController 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.
@RegistJSON
@RegistCSV,
@RegistTXT
@RegistBIN
@ReferenceLatestData
@DataCount
ユーザ変数 : '@'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.
@RegistJSON
@RegistCSV,
@RegistTXT
@RegistBIN

@ReferenceLatestData
@DataCount
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.

2.3.2.1. @RegistJSON

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 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, 以下のオプションを指定してください.

表 2-3 @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	-	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	-	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	-	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.

(*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．

2.3.2.2. @RegistCSV

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 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，以下のオプションを指定してください．

表 2-4 @RegistCSV 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	-	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	-	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	-	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．

2.3.2.3. @RegistTXT

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 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，以下のオプションを指定してください．

表 2-5 @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	-	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.

2.3.2.4. RegistBIN

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 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, 以下のオプションを指定してください.

表 2-6 @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.

2.3.2.5. @ReferenceLatestData

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"@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, リソース_JSON の最新データを文字列(BSTR)として取得できます.

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

オプションを指定してください.

表 2-7 @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.
Extension	-	以下を指定. ・json 省略時は json が指定されたものとして扱います.
Select	-	QUERY の$select パラメータを指定します. <選択key>で指定されたフィールドのデータのみを返します. ・<選択key>は, JSON でのname, XML での要素名, 属性名に相当し, 登録データ内の任意のkeyを指定できます. フィールドの階層は「. 」で表現します. ・<選択key>は「, 」区切りで複数指定できます. ・本サービスの管理データである, _date/_resource_path/_data は<選択key>に使用できません. 例)ensor.id, sensor.name, sensor.data.temp

2.3.2.6. @DataCount

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"@DataCount"を指定して使用します. get することで"DataCount"コマンドと同様に, リソース_JSON の HIT データ数(I4)を取得できます.

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, 以下のオプションを指定してください.

表 2-8 @DataCount のオプション文字列

設定項目	必須	備考
ResourcePath	○	リソースパス(/$all 利用可)を指定します. このオプションの詳細は 4.1.を参照してください.
Filter	-	QUERY の$filter パラメータを指定します. <filter 条件>に一致するもののみを返すよう, 結果を限定します. <filter 条件>は,「プロパティ名 演算子 条件値」とし, and or で複数定義可能です. 使用可能な演算子, プロパティ名は後述します. このオプションの詳細は4.2.を参照してください.

2.4. 変数一覧

IoT Platform 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	リソース_JSON への JSON 文字列のデータ登録(蓄積)を行います.	-	○
@RegistCSV	VT_BSTR	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	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	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	リソース_JSON の最新データ参照を行います.	○	-
@DataCount	VT_I4	リソース_JSON 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. Controller クラス

表 3-1 CaoController::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	ユーザ指定された URL パス, クエリを付与して PUT, GET, DELETE, POST を実行します.	-	17
データ登録/転送	RegistJSON	リソース_JSON への JSON 文字列のデータ登録(蓄積)を行います.	@RegistJSON	19
	RegistCSV	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	20
	RegistTXT	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	21
	RegistBIN	リソース_JSON へのバイナリデータ登録(蓄積)を行います.	@RegistBIN	22
リソースデータの参照	ReferenceLatestData	リソース_JSON の最新データ参照を行います.	@ReferenceLatestData	23
	ReferencePastData	リソース_JSON の過去データ(指定日時)参照を行います.	-	24
リソースデータの検索	RetrieveData	リソース_JSON のデータ検索を行います.	-	25
	DataCount	リソース_JSON の HIT データ数を取得します.	@DataCount	26
リソースデータの更新	UpdateJSON	リソース_JSON の JSON 文字列の更新を行います.	-	27
	UpdateCSV	リソース_JSON の CSV 文字列の更新を行います.	-	28
	UpdateTXT	リソース_JSON の TXT 文字列の更新を行います.	-	29
	UpdateBIN	リソース_JSON のバイナリデータの更新を行います.	-	30
リソースデータの削除	DeleteData	リソース_JSON のデータ削除を行います.	-	31

3.1.1. 汎用 REST

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

富士通 IoT Platform 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 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 へのデータ登録/転送

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

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>)

<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 = <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 = <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.

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

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>)

<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.

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

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　=　<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. CaoController::Execute("RegistBIN") コマンド

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. リソースデータの参照

3.1.3.1. CaoController::Execute("ReferenceLatestData") コマンド

リソース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>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<リソースパス>(BSTR)
必須要素

第 2 要素　=　<拡張子>(BSTR)
省略可能要素
以下を指定.
・json
省略時は json が指定されたものとして扱います.

第 3 要素　=　< QUERY の$select パラメータ>(BSTR)
省略可能要素
<選択 key>で指定されたフィールドのデータのみを返します.
・<選択 key>は, JSON での name, XML での要素名, 属性名に相当し, 登録データ内の任意の key を指定できます. フィールドの階層は「.」で表現します.
・<選択 key>は「,」区切りで複数指定できます.
・本サービスの管理データである, _date/_resource_path/_data は<選択 key>に使用できません.

戻り値　:　[out] (BSTR)
参照したデータ.

3.1.3.2. CaoController::Execute("ReferencePastData") コマンド

登録日時が指定日時と一致する, リソース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>)

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

第 1 要素 = <リソースパス>(BSTR)
必須要素
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 = <拡張子>(BSTR)
省略可能要素
以下を指定.
・json
省略時は json が指定されたものとして扱います.
第 4 要素 = < select 条件>(BSTR)
省略可能要素
<選択 key>で指定されたフィールドのデータのみを返します.
・<選択 key>は, JSON での name, XML での要素名, 属性名に相当し, 登録データ内の任意の key を指定できます. フィールドの階層は「.」で表現します.
・<選択 key>は「,」区切りで複数指定できます.
・本サービスの管理データである, _date/_resource_path/_data は<選択 key>に使用できません.

戻り値 : [out] (BSTR)
参照したデータ.

3.1.4. リソースデータの検索

3.1.4.1. CaoController::Execute("RetrieveData") コマンド

指定条件に一致するリソース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) 検索結果として得られるデータの順番は, <リソースパス>および<登録日時>でソートします.

書式　RetrieveData (<Data>)

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

第 1 要素 = <リソースパス(/$all 利用可) >(BSTR)
必須要素
詳細は 4.1.を参照してください.

第 2 要素 = <拡張子>(BSTR)
省略可能要素
以下を指定.
・json
省略時は json が指定されたものとして扱います.

第 3 要素 = <filter 条件>(BSTR)
省略可能要素
<filter 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 4.2.を参照してください.

第 4 要素 = < top 条件>(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 1000 件です.

第 5 要素 = < skip 条件>(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, 第一ソート key:<リソースパス>, 第二ソート key:<登録日時>でソートします.

第 6 要素 = < select 条件>(BSTR)
省略可能要素

<選択 key>で指定されたフィールドのデータのみを返します.
・<選択 key>は, JSON での name, XML での要素名, 属性名に相当し, 登録データ内の任意の key を指定できます. フィールドの階層は「.」で表現します.
・<選択 key>は「,」区切りで複数指定できます.
・本サービスの管理データである, _date/_resource_path/_data は<選択 key>に使用できません.

戻り値 : [out] (BSTR)
一致したリソースデータ.

3.1.4.2. CaoController::Execute("DataCount") コマンド

指定条件に一致するリソース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.
(*1) 検索結果として得られるデータの順番は, <リソースパス>および<登録日時>でソートします.

書式 DataCount (<Data>)

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

第 1 要素 = <リソースパス(/$all 利用可) >(BSTR)
必須要素
詳細は 4.1.を参照してください.
第 2 要素 = <filter 条件>(BSTR)
省略可能要素
<filter 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 4.2.を参照してください.

戻り値 : [out] (I4)
一致したリソースデータ数.

3.1.5. リソースデータの更新

3.1.5.1. CaoController::Execute("UpdateJSON") コマンド

指定条件に一致するリソースに登録済みのデータを JSON 文字列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.
(*1) 同じ登録日時のデータが複数存在した場合, 1 件のみ更新されます. (どのデータになるかは不定)

書式　UpdateJSON (<Data>)

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

第 1 要素 = <リソースパス>(BSTR)
必須要素
第 2 要素 = <更新対象データの登録日時>(BSTR)
必須要素
第 3 要素 = <送信する JSON 文字列>(BSTR)
必須要素
第 4 要素 = <更新後の登録日時>(BOOL)
省略可能要素
・省略時は登録日時を更新しません.
・既に指定登録日時のデータが存在するかどうかは
チェックせず, 上書きを行います.

3.1.5.2. CaoController::Execute("UpdateCSV") コマンド

指定条件に一致するリソースに登録済みのデータを CSV 文字列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.
(*1) 同じ登録日時のデータが複数存在した場合, 1 件のみ更新されます. (どのデータになるかは不定)

書式　UpdateCSV (<Data>)

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

第 1 要素 = <リソースパス>(BSTR)
必須要素

第 2 要素 = <更新対象データの登録日時>(BSTR)
必須要素

第 3 要素 = <送信する CSV 文字列>(BSTR)
必須要素

第 4 要素 = <更新後の登録日時>(BOOL)
省略可能要素
・省略時は登録日時を更新しません.
・既に指定登録日時のデータが存在するかどうかはチェックせず, 上書きを行います.

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 false が指定されたものとします.

3.1.5.3. CaoController::Execute("UpdateTXT") コマンド

指定条件に一致するリソースに登録済みのデータを TXT 文字列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.
(*1) 同じ登録日時のデータが複数存在した場合, 1 件のみ更新されます. (どのデータになるかは不定)

書式 UpdateTXT (<Data>)

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

第 1 要素 = <リソースパス>(BSTR)
必須要素
第 2 要素 = <更新対象データの登録日時>(BSTR)
必須要素
第 3 要素 = <送信する TXT 文字列>(BSTR)
必須要素
第 4 要素 = <更新後の登録日時>(BOOL)
省略可能要素
・省略時は登録日時を更新しません.
・既に指定登録日時のデータが存在するかどうかは
チェックせず, 上書きを行います.

3.1.5.4. CaoController::Execute("UpdateBIN") コマンド

指定条件に一致するリソースに登録済みのデータをバイナリ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.
(*1) 同じ登録日時のデータが複数存在した場合, 1 件のみ更新されます. (どのデータになるかは不定)

書式　UpdateBIN (<Data>)

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

第 1 要素 = <リソースパス>(BSTR)
必須要素
第 2 要素 = <更新対象データの登録日時>(BSTR)
必須要素
第 3 要素 <Content-Type に指定する MIME-TYPE>(BSTR)
必須要素
第 4 要素 = <送信するバイナリ>(ARRAY|UI1)
必須要素
第 5 要素 = <更新後の登録日時>(BOOL)
省略可能要素
・省略時は登録日時を更新しません.
・既に指定登録日時のデータが存在するかどうかはチェックせず, 上書きを行います.
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.
省略時は Body データが無圧縮とみなします.
・gz

3.1.6. リソースデータの削除

3.1.6.1. CaoController::Execute("DeleteData") コマンド

指定条件に一致するリソース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>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<リソースパス>(BSTR)
必須要素
第 2 要素　=　<filter 条件>(BSTR)
必須要素
<filter 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 4.2.を参照してください.

3.1.7. リソースの登録

3.1.7.1. CaoController::Execute("RegistMetadata") コマンド

新規リソースの登録をします.
不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RegistMetadata (<Data>)

<Data>　:　[in] (ARRAY | BSTR)

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)
省略可能要素
詳細は 4.3.1 を参照してください
省略時は, リソースのみ登録

3.1.8. リソースのメタデータ参照

3.1.8.1. CaoController::Execute("ReferenceMetadata") コマンド

指定条件に一致するリソースに登録されている全てのメタ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　ReferenceMetadata (<Data>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<リソースパス(/$all)>（BSTR）
必須要素
詳細は 4.1.を参照してください.

第 2 要素　=　< top 条件>（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 1000 件です.

第 3 要素　=　< skip 条件>（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, 第一ソート key:<リソースパス>, 第二ソート key:<登録日時>でソートします.

戻り値　:　[out] (BSTR)
参照したリソースデータ.

3.1.9. リソースのメタデータ更新

3.1.9.1. CaoController::Execute("UpdateMetadata") コマンド

リソースに対して JSON 形式で対象リソースのメタデータを，更新データで全て上書きします．
不正な引数を指定した場合は E_INVALIDARG が発生します．

書式　UpdateMetadata (<Data>)

<Data>　:　[in] (ARRAY|BSTR)

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
詳細は 4.3.1 を参照してください

3.1.10. リソースの削除

3.1.10.1. CaoController::Execute("DeleteMetadata") コマンド

指定したリソースを削除します．
不正な引数を指定した場合は E_INVALIDARG が発生します．

書式　DeleteMetadata (<Data>)

<Data>　:　[in](BSTR)
<リソースパス> 必須要素

3.1.11. アクセスコードの登録

3.1.11.1. CaoController::Execute("RegistAccessCodeData") コマンド

リソースに対して，アクセスコードを新規登録します.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　RegistAccessCodeData (<Data>)

<Data>　:　[in] (ARRAY |BSTR)

第 1 要素　=　<アクセスコード>（BSTR）
必須要素
登録するアクセスコード名

第 2 要素　=　<送信する JSON 文字列>（BSTR）
必須要素
詳細は 4.3.2 を参照してください

3.1.12. アクセスコードの参照

3.1.12.1. CaoController::Execute("ReferenceAccessCodeData") コマンド

指定条件に一致するリソースに登録されている全てのアクセスコードを参照します.
不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　ReferenceAccessCodeData (<Data>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<アクセスコード>(BSTR)
必須要素
第 2 要素　=　<filter 条件>(BSTR)
省略可能要素
<filter 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 4.2.を参照してください.
第 3 要素　=　< top 条件>(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 1000 件です.
第 4 要素　=　< skip 条件>(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, 第一ソート key:<リソースパス>, 第二ソート key:<登録日時>でソートします.

戻り値　:　[out] (BSTR)
一致したアクセスコード情報.

3.1.12.2. CaoController::Execute("DataCountAccessCodeData") コマンド

指定条件に一致するリソースに登録されているアクセスコード数を取得します.
不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　DataCountAccessCodeData (<Data>)

<Data>　:　[in] (ARRAY|BSTR)

第 1 要素　=　<アクセスコード>(BSTR)
必須要素
第 2 要素　=　<filter 条件>(BSTR)
省略可能要素
<filter 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 4.2.を参照してください.

戻り値　:　[out] (I4)
一致したアクセスコード数.

3.1.13. アクセスコードの更新

3.1.13.1. CaoController::Execute("UpdateAccessCodeData") コマンド

アクセスコードの更新をします.
不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　UpdateAccessCodeData (<Data>)

<Data>　:　[in] (ARRAY| BSTR)

第 1 要素　=　<アクセスコード>(BSTR)
必須要素.
第 2 要素　=　<送信する JSON 文字列>(BSTR)
必須要素
詳細は 4.3.2 を参照してください

3.1.14. アクセスコードの削除

3.1.14.1. CaoController::Execute("DeleteAccessCodeData") コマンド

指定したアクセスコードの削除を行います.

不正な引数を指定した場合は E_INVALIDARG が発生します.

書式　DeleteAccessCodeData (Data)

<Data>　:　[in] (BSTR)
<アクセスコード>(BSTR)
必須要素

3.1.15. イベントの登録

3.1.15.1. CaoController::Execute("RegistEventData") コマンド

指定したリソースデータに対して新規イベント情報を登録することができます.

不正な引数を指定した場合は E_INVALIDARG が発生します

書式　RegistEventData (<Data>)

<Data>　:　[in] (BSTR)
<送信する JSON 文字列>
必須要素
詳細は 4.3.3 を参照してください

戻り値　:　[out] (BSTR)
登録されたイベント情報のイベント ID.

3.1.16. イベント情報の参照

3.1.16.1. CaoController::Execute("ReferenceEventData") コマンド

イベント情報の参照を行います.

不正な引数を指定した場合は E_INVALIDARG が発生します

書式　ReferenceEventData (<Data>)

<Data>　:　[in] (ARRAY|VARIANT)

第 1 要素　=　<イベント ID>(BSTR)
省略可能要素
※第 2～4 要素の条件でイベント情報を参照したい場合は, null を指定

第 2 要素　=　<filter 条件>(BSTR)
省略可能要素
<filter 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 4.2.を参照してください.

第 3 要素　=　< top 条件>(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 1000 件です.

第 4 要素　=　< skip 条件>(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, 第一ソート key:<リソースパス>, 第二ソート key:<登録日時>でソートします.

戻り値　:　[out] (BSTR)
一致したイベント情報.

3.1.17. イベント情報の更新

3.1.17.1. CaoController::Execute("UpdateEventData") コマンド

指定したイベント ID のイベント情報の更新をします.
不正な引数を指定した場合は E_INVALIDARG が発生します

書式　UpdateEventData(<Data>)

<Data>　:　[in] (ARRAY| BSTR)
第 1 要素　=　<イベント ID>(BSTR)
必須要素.
更新したいイベント ID
第 2 要素　=　<送信する JSON 文字列>
必須要素
詳細は 4.3.3 を参照してください

3.1.18. イベントの削除

3.1.18.1. CaoController::Execute("DeleteEventData") コマンド

指定したイベント ID のイベント情報を削除します.
不正な引数を指定した場合は E_INVALIDARG が発生します

書式　DeleteEventData (Data)
<Data>　:　[in] (BSTR)
<イベント ID>
必須要素

4. 補足

4.1. <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.

4.2. filter 条件について

4.2.1. filter 条件の演算子

演算子	説明	例
eq	等号	Owner eq 'Tom'
ne	不等号	Owner ne null
gt	より大きい	Floor1.Value gt 1000
ge	以上	Floor1.Value ge 1000
lt	より小さい	Floor1.Value lt 1000
le	以下	Floor1.Value le 1000
and	論理積	Floor1.Value ge 1000 and Owner eq 'Tom'
or	論理和	Id eq 2 or Id eq 1

1. null は, 値が存在しないことを示します.
2. データの name が階層構造になっている場合は, <name>.<name>で表現します.
3. (A eq 1 and B eq 1)or(A eq 2 and B eq 2)のような表現も可能です.
   - ただしこの場合, ()内に()は定義できません. 例えば, ((A eq 1 and B eq 1)or(A eq 2 and B eq 2))and(C eq 1)はエラーとなります.
4. 正規表現によるマッチングには対応していません.
5. 文字列はシングルクォートで囲みます. シングルクォートで囲まないものは数値とみなします.

6. JSON 配列の内容を指定する場合は, <name>.<配列 index>で表現します. 配列 index は数字です.
   例:

検索対象データ	{
	"Owners":[
	"Taro",
	"Jiro"
	]
	}

filter 条件指定例	Owners.0 eq 'Taro'

- 配列内にオブジェクト構造があり, かつ name が数字のみで構成されている場合, 配列 index 指定と name の識別ができない場合があります. 下記例を参考に配列 index と name の双方を指定してください.
  検索対象データ例:

① {"data":[{"0":"Taro"},{"0":"Jiro"}]}
② {"data":[{"0":"Jiro"},{"0":"Taro"}]}
③ {"data":{"0":"Taro"}}

検索例:

filter 指定条件	適合するデータ
data.0 eq 'Taro'	①, ②, ③
data.1.0 eq 'Taro'	②

4.2.2. filter 条件に利用可能なプロパティ名

プロパティ名	説明	備考
_date	登録日時	検索対象データの登録日時(*1) ※_date で条件指定する登録日時については シングルクォートを使って囲みません.
任意の name	登録データに含まれる任意の name	URI の非予約文字 (「半角英数字」,「-」,「.」,「_」,「~」以外)は パーセントエンコードしてください.

(*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. 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」を添えてください. 本サービスがレスポンスに格納する場合, UTC を用います.

例)
下記の JSON 構造をしたリソースパスに対して,
赤枠で囲んだ部分を上から, 第 1~3 要素と呼ぶ場合.


```
[
    {
        "_data": {
            "Key3": "Value3"
        },
        "_date": "20180730T044740.351Z",
        "_resource_path": "Test/ABC"
    },
    {
        "_data": {
            "Key": "Value"
        },
        "_date": "20180730T044607.267Z",
        "_resource_path": "Test/ABC"
    },
    {
        "_data": {
            "Key": "Value"
        },
        "_date": "20180730T044549.801Z",
        "_resource_path": "Test/ABC"
    }
]
```


・第 1 要素の情報のみ取得する場合は, filter 条件に下記を指定します.
「_date eq _date eq 20180730T044740.351Z」もしくは, 「Key3 eq 'Value3'」

4.2.3. アクセスコード参照・イベント情報の参照で使用可能な filter 条件について

filter 条件演算子

演算子	説明	例
eq	等号	Owner eq 'Tom'

filter 条件のプロパティ名

プロパティ名	説明	備考
_resource_path	リソースパス	「/」も含めてパーセントエンコードは不要です

filter 条件のサポート関数

Function	説明	例
bool startwith(string p0,string p1)	前方一致	/abc* ?$filter=startswith(_resource_path,'hoge') eq true

1. false 指定は未対応です.
2. $filter 条件直下で_resource_path eq hoge と指定されたときは完全指定となり startswith()を利用して指定された際は前方一致指定となります.

(2018 年 7 月現在)

4.3. 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

4.3.1. リソース制御時

Parameters	形式	M/O	説明	最大長
resource	-	M	開始タグ	-
retention_period	string	O(*2)	リソースデータの保存期間(日) ・設定されていない場合，保存期間は 1 日とします． ・値の範囲は，1～9999	9999
fwd_info		O(*3)	転送先情報	-
http	-	M	HTTP の転送情報	-
method	-	M	"GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS", "TRACE"のいずれか	7 文字
uri	string	M	URI. "http://～"または"https://～"	256 文字
basic_auth_id	string	O	Basic 認証用の ID	20 文字
basic_auth_pass	string	O	Basic 認証用のパスワード	20 文字
header_fields	- (配列)	O	HTTP ヘッダ (配列の要素は最大 10 個)	-
field_name	string	M	ヘッダフィールド名.":"は含みません．	20 文字
field_value	string	M	上記ヘッダフィールドに格納する値	512 文字

(*1) M：必須，O：オプション．各子要素のM/O は，親要素を設定した場合の要否を表します．

(*2) リソース_JSON，リソース_Binary の場合に限り有効です．

(*3) 対象リソースが JSON 形式の転送リソースの場合のみ設定できます．

入力例は，

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

11 章 11.1 リソースの登録例をご参照ください．

4.3.2. アクセスコード制御時

Parameters	形式	M/O	説明	最大長
access_code	-	M	開始タグ	-
permissions	-	M	権限情報	-
ip_filter	string (配列)	O	サービスポータルで定義したアクセス制限(アクセスコード)設定の IP アドレス範囲情報を指定してください. 未定義の IP アドレス範囲情報を設定した場合, エラーレスポンス(400 Bad Request)を返却します. ・["開始 IP アドレス", "終了 IP アドレス"]の形式で2つの IP アドレスを指定できます. ・1 つの IP アドレスを設定する場合, 開始 IP アドレスと終了 IP アドレスを同じアドレスにしてください. ・「開始 IP アドレス＞終了 IP アドレス(IP アドレスを32bit とみなして比較した場合)」となる設定はできませ	35 文字
resource_operations	- (配列)	M	リソースに紐づく権限情報	
resource_path	string	M	リソースパス	128 文字
operations	string (配列)	M	"hierarchy_get", "hierarchy_put", "create", "read", "update", "delete", "list"のいずれか. 配列で複数指定できます.	72 文字
certification_info	-	O	クライアント情報	
certification	string	M	クライアント証明書 ・PEM 形式 「-----BEGIN CERTIFICATE-----」, 「-----END CERTIFICATE-----」を含めてください. また, 改行コードは「¥n」として登録してください.	10000 文字
certificate_usage	string	M	クライアント証明書の利用用途. 以下を指定してください. ・auth:クライアント認可で使用する.	4 文字
protocols	string (配列)	O	プロトコル指定情報 "http", "https", "mqtt", "mqtts" のいずれか. 配列で複数指定できます	29 文字

(*1) M:必須, O:オプション. 各子要素の M/O は, 親要素を設定した場合の要否を表します.

入力例は,
https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/userguide.pdf
12 章 12.1 アクセスコードの登録例をご参照ください.

4.3.3. イベント制御時

Parameters	形式	M/O	説明	最大長
event	-	M	開始タグ	-
conditions	-	M	イベント条件	-
targets	-	M	対象	-
resource_path	string	M	対象のリソースパス バイナリデータへの操作の場合は対象リソースパスを"_bin/"から設定してください.	128 文字
operations	string (配列)	M	対象のリソースデータ操作 （アクセスコードのアクセス権とは異なります.） ・通常のリソースの場合，["create","update"]を指定してください. ・リソース_Binary の場合，"create"を指定してください.	33 文字
read_access_code	string	M	対象リソースパスの read 権または hierarchy_get 権を持つアクセスコード	48 文字
notification_condition	-	O	通知条件	-
start_time	日時または時刻	O(*2)	開始日時または，時刻(*3)	20 文字
end_time	日時または時刻	O(*4)	終了日時または，時刻(*3)	20 文字
body_conditions	-	O	データボディ条件 Binary リソースに対しては設定不可	-
path_type	string	M	path の形式 "JSONPath"のみ指定できます	8 文字
path	string	M	ボディ要素を指定するためのパス	1902 文字
comparing_operator	string	M	比較演算子 "eq"(＝)，"ne"(≠)，"gt"(>)，"ge"(≧)，"lt"(<)，"le"(≦), "substring_of"(部分一致)のいずれか. (*5)	12 文字

					value						string または 数値		比較対象の値.	— (*7)
			awake_condition								-	O	イベント抑止解除条件. Binary リソースに対しては設定できません. ※本設定を行うと, notification_condition を満たしイベント通知実施後, イベント抑止状態になります. イベント抑止状態を解除するための条件をここで設定してください.	-
					body_conditions						-	M	データボディ条件 Binary リソースに対しては設定不	
						path_type					string	M	path の形式 "JSONPath"のみ指定できます	8 文字
						path					string	M	ボディ要素を指定するためのパス	1902 文字
						comparing_operator					string	M	比較演算子 "eq"(＝), "ne"(≠), "gt"(>), "ge"(≧), "lt"(<), "le"(≦), "substring_of"(部分一致)のいずれか. (*5)	12 文字
						value					string または 数値		比較対象の値.	— (*7)
	notification										—(*6)	M	通知内容	-
		http									-	O	HTTP 通知設定	-
			method								string	M	"GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS", "TRACE"のいずれか.	7 文字
			uri								string	M	URI. "http://～"または "https://～"	256 文字
			basic_auth_id								string	O	Basic 認証用の ID	20 文字
			basic_auth_pass								string	O	Basic 認証用のパスワード	20 文字
			header_fields								-	O	HTTP ヘッダ	-

					(配列)		(配列の要素は最大 10 個)	
				field_name	string	M	ヘッダフィールド名. ":"は含みません.	20 文字
				field_value	string	M	上記ヘッダフィールドに格納する値	512 文字
			body		string	O	ボディに格納する値 省略時は, イベントのトリガとなったリソースデータ本体・イベント ID・イベント発生日時・対象リソースパス	1024 文字
		smtp			-	O	SMTP 通知設定	-
			send_to		string	M	通知先 E メールアドレス	256 文字
			subject		string	O	件名	256 文字
			body		string	M	本文	140 文字

(*1) M:必須, O:オプション. 各子要素のM/O は, 親要素を設定した場合の要否を表します.
(*2) start_time とend_time の片方のみを設定することはできません. また, 両設定値は日時または, 時刻のどちらかに揃える必要があります.
(*3) 日時指定の場合は, ISO8601 に従ってください(20141225T103612Z など). また, 精度は秒まで指定可能です. 時刻指定の場合はISO8601 から年月日の情報を削ったものとなります(T103612Z など). 精度は同様に秒までです. タイムゾーン指定は「±hhmm」形式で省略時は「Z」を添えてください.
(*4) start_time とend_time の片方のみを指定することはできません. また, 両設定値は日時または, 時刻のどちらかに揃える必要があります.
(*5) value が数値の場合は, eq, ne, gt, ge, lt, le が指定できます. value が文字列の場合はeq, ne,
substring_of が指定できます. 文字列の比較においては大文字小文字の区別を行います.
(*6) notification 配下には, http か smtp いずれかの設定が必要です.
(*7) 文字列の場合, 1～128 文字
整数の場合, -2, 147, 483, 648 ～2, 147, 483, 647
実数の場合, 整数部分 10 桁, 小数部分 5 桁

入力例は,
https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v5/userguide.pdf
13 章 13.1 イベントの登録例をご参照ください.

4.4. 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

5. サンプルプログラム

以下に C# のサンプルを示します.

5.1. 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.

List 5-1

```
…(略)…
using ORiN2.ManagedCAO;

namespace IoT Platform_Sample
{
    public  class Sample
    {
          private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
          private readonly string TENANT_ID = "TenantID";
          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 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, ACCESS_CODE);

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

               // オプション文字列
```

```
                var registBinOpt = string.Format("{0}, MimeType=image/jpeg, Retain=true",
RESOURCE_OPT);
                var dataCountOpt = "ResourcePath=Test/$all";

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

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

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

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

                systemVariable = m_CaoController.AddVariable("@RegistBIN", registBinOpt);
                userVariable = m_CaoController.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;
                }

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

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

                // @DataCount
                systemVariable = m_CaoController.AddVariable("@DataCount", dataCountOpt);
                userVariable = m_CaoController.AddVariable("DataCount",
string.Format("Type=@DataCount,{0}", dataCountOpt));

                var datacount = (int)systemVariable.Value;
                datacount = (int)userVariable.Value;
            }
        }
    }
```

5.2. 汎用 REST

汎用 REST の使用例を示します.

List 5-2

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

namespace IoT Platform_Sample
{
    public  class Sample
    {

        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly string TENANT_ID = "TenantID";
        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 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, ACCESS_CODE);


            if (m_CaoController == null)
            {
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
"CaoProv.FUJITSU.IoTPlatform", null, opstionStr);
            }
            var paramList = new List<object>();
```

```
                // 汎用REST
                paramList.Add("PUT");
                paramList.Add(RESOUCE_PATH);
                var restResult = m_CaoController.Execute("REST", paramList.ToArray());
            }
        }
    }
```

5.3. データ制御

リソース_JSON へのデータ登録/転送, リソースデータの参照, リソースデータの検索, リソースデータの更新, リソースデータの削除 の使用例を示します.

List 5-3

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

namespace IoT Platform_Sample
{
    public  class Sample
    {

       private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
       private readonly string TENANT_ID = "TenantID";
       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 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, ACCESS_CODE);


           if (m_CaoController == null)
           {
               // コントローラに接続
```

```
                m_CaoController = m_CaoWorkspace.AddController("Test",
"CaoProv.FUJITSU.IoTPlatform", null, opstionStr);
            }
            // 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_CaoController.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_CaoController.Execute("RegistCSV", paramList.ToArray());

            // RegistTXT
            paramList.Clear();
            paramList.Add(RESOUCE_PATH);
            paramList.Add("testTxt");
            m_CaoController.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_CaoController.Execute("RegistBIN", paramList.ToArray());
            }

            // リソースデータの参照
            // ReferenceLatestData
            paramList.Clear();
            paramList.Add(RESOUCE_PATH);
            var referenceLatestData = m_CaoController.Execute("ReferenceLatestData",
paramList.ToArray());

            // ReferencePastData
            paramList.Clear();
            paramList.Add(RESOUCE_PATH);
            paramList.Add(REGIST_DATE);
            var referencePastData = m_CaoController.Execute("ReferencePastData",
paramList.ToArray());

            // リソースデータの検索
            // RetrieveData
            paramList.Clear();
            paramList.Add(string.Format("{0}/$all", RESOUCE_PATH));
            paramList.Add("json");
            paramList.Add(null);
            paramList.Add(100);
            var retrieveData = m_CaoController.Execute("RetrieveData", paramList.ToArray());

            // DataCount
            paramList.Clear();
            paramList.Add(string.Format("{0}/$all", RESOUCE_PATH));
            paramList.Add("_date gt 20180830T082135.645Z");
            var dataCount = m_CaoController.Execute("DataCount", paramList.ToArray());

            // リソースデータの更新
            // UpdateJSON
            paramList.Clear();
```

```
paramList.Add(RESOUCE_PATH);
paramList.Add(REGIST_DATE);
paramList.Add("{¥"test¥":2}");

m_CaoController.Execute("UpdateJSON", paramList.ToArray());

// UpdateCSV
paramList.Clear();
csvStr = @"name,age,tel
                田中,31,012-345-6789
                鈴木,42,098-765-4321";
paramList.Add(RESOUCE_PATH);
paramList.Add(REGIST_DATE);
paramList.Add(csvStr);
m_CaoController.Execute("UpdateCSV", paramList.ToArray());

// UpdateTXT
paramList.Clear();
paramList.Add(RESOUCE_PATH);
paramList.Add(REGIST_DATE);
paramList.Add("testTxt2");
m_CaoController.Execute("UpdateTXT", paramList.ToArray());

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

// リソースデータの削除
// DeleteData
paramList.Clear();
```

```
                paramList.Add(RESOUCE_PATH);
                paramList.Add(string.Format("_date gt {0}", REGIST_DATE));
                m_CaoController.Execute("DeleteData", paramList.ToArray());
            }
        }
    }
```

5.4. リソース制御

リソースの登録, リソースのメタデータ参照, リソースのメタデータ更新, リソースの削除 の使用例を示します.

List 5-4

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

namespace IoT Platform_Sample
{
    public  class Sample
    {

        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly string TENANT_ID = "TenantID";
        private readonly string ACCESS_CODE = "AccessCode";
        private readonly string EVENT_ACCESS_CODE = "TemperatureCDL";
        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 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, ACCESS_CODE);

            if (m_CaoController == null)
            {
```

```
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
"CaoProv.FUJITSU.IoTPlatform", null, opstionStr);
            }
            var paramList = new List<object>();
            var paramStrList = new List<string>();

            // リソース制御
            // リソースの登録
            // RegistMetadata
            paramStrList.Add(string.Format("{0}/child", RESOUCE_PATH));
            paramList.Add("{¥"resource¥": {¥"retention_period¥" : 2}}");
            m_CaoController.Execute("RegistMetadata", paramStrList.ToArray());

            // リソースのメタデータ参照
            // ReferenceMetadata
            paramList.Clear();
            paramList.Add(string.Format("{0}/$all", RESOUCE_PATH));
            paramList.Add(100);
            var referenceMetadata = m_CaoController.Execute("ReferenceMetadata",
paramStrList.ToArray());

            // リソースのメタデータ更新
            // UpdateMetadata
            paramStrList.Clear();
            paramStrList.Add(string.Format("{0}/child", RESOUCE_PATH));
            paramStrList.Add("{¥"resource¥": {¥"retention_period¥" : 3}}");
            //{"resource": {"retention_period" : 3}}
            m_CaoController.Execute("UpdateMetadata", paramStrList.ToArray());

            // リソースの削除
            // DeleteMetaData
            m_CaoController.Execute("DeleteMetaData", string.Format("{0}/child",
RESOUCE_PATH));
        }
    }
}
```

5.5. アクセスコード制御

アクセスコードの登録, アクセスコードの参照, アクセスコードの更新, アクセスコードの削除 の使用例を示します.

List 5-5

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

namespace IoT Platform_Sample
{
    public  class Sample
    {

        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly string TENANT_ID = "TenantID";
        private readonly string ACCESS_CODE = "AccessCode";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, ACCESS_CODE);

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

            var paramList = new List<object>();
```

```
            var paramStrList = new List<string>();

            // アクセスコード制御
            // アクセスコード付与用のリソースパス作成
            var accessCodeRegistResoucePath = "Test/child";
            paramStrList.Add(accessCodeRegistResoucePath);
            paramStrList.Add("{¥"resource¥": {¥"retention_period¥" : 2}}");
            m_CaoController.Execute("RegistMetadata", paramStrList.ToArray());

            // 登録アクセスコード
            var registAccessCode = "AccessCodeTest";

            // アクセスコードの登録
            // RegistAccessCodeData
            paramStrList.Clear();
            paramStrList.Add(registAccessCode);
            var registAccessCodeJson = "{¥"access_code¥":{ ¥"permissions¥":
{¥"resource_operations¥": [{¥"operations¥": [¥"update¥",¥"read¥"],¥"resource_path¥":
¥"Test/child¥"}]}}}";
            paramStrList.Add(registAccessCodeJson);
            m_CaoController.Execute("RegistAccessCodeData", paramStrList.ToArray());

            // アクセスコードの参照
            // ReferenceAccessCodeData
            paramList.Clear();
            paramList.Add(registAccessCode);
            var referenceAccessCodeData = m_CaoController.Execute("ReferenceAccessCodeData",
paramList.ToArray());

            // DataCountAccessCode
            paramStrList.Clear();
            paramStrList.Add(registAccessCode);
            var dataCountAccessCode = m_CaoController.Execute("DataCountAccessCodeData",
paramStrList.ToArray());

            // アクセスコードの更新
            // UpdateAccessCodeData
```

```
                paramStrList.Clear();
                paramStrList.Add(registAccessCode);
                paramStrList.Add(registAccessCodeJson);
                m_CaoController.Execute("UpdateAccessCodeData", paramStrList.ToArray());

                // アクセスコードの削除
                // DeleteAccessCode
                m_CaoController.Execute("DeleteAccessCodeData", registAccessCode);

                // アクセスコード付与用のリソース削除
                m_CaoController.Execute("DeleteMetaData", accessCodeRegistResoucePath);
            }
        }
    }
```

5.6. イベント制御

イベントの登録 , イベント情報の参照 , イベント情報の更新, イベントの削除 の使用例を示します.

List 5-6

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

namespace IoTPlatform_Sample
{
    public  class Sample
    {
        private readonly string BASE_URL = @"http://<zone>.fujitsu.com";
        private readonly string TENANT_ID = "TenantID";
        private readonly string EVENT_ACCESS_CODE = "TemperatureCDL";

        private CCaoEngine m_CaoEngine = null;
        private CCaoWorkspaces m_CaoWorkspaces = null;
        private CCaoWorkspace m_CaoWorkspace = null;
        private CCaoController m_CaoController = 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},TenantID={1},AccessCode={2}",
BASE_URL, TENANT_ID, EVENT_ACCESS_CODE);

            if (m_CaoController == null)
            {
                // コントローラに接続
                m_CaoController = m_CaoWorkspace.AddController("Test",
"CaoProv.FUJITSU.IoTPlatform", null, opstionStr);
            }
            var paramList = new List<object>();
            var paramStrList = new List<string>();
            // イベント制御
            // TODO IoT Platformダッシュボードにて
            // リソース：「sensors」「sensors/temperature」 を作成
            // アクセスコード：「TemperatureCDL」, 対象リソースは「sensors」,「CDL」権限付与
            // アクセスコード:「TemperatureRU」, 対象リソースは「sensors/temperature」,「RU」
権限付与

            // イベント登録
            // RegistEventData
            var registEventDataJson = "{¥"event¥": {¥"notification¥": {¥"smtp¥": {¥"subject¥":
¥"温度が３０度超えました¥",¥"body¥": ¥"温度が３０度を超えています. 確認してください.
¥",¥"send_to¥": ¥"hoge@piyo.fujitsu.com¥"}},¥"conditions¥": {¥"targets¥":
{¥"read_access_code¥": ¥"TemperatureRU¥",¥"operations¥":
[¥"create¥",¥"update¥"],¥"resource_path¥":
¥"sensors/temperature¥"},¥"notification_condition¥": {¥"body_conditions¥": {¥"path¥":
¥"sensors/temperature¥",¥"path_type¥": ¥"JSONPath¥",¥"comparing_operator¥":
¥"ge¥",¥"value¥": 30.0}}}}}";
            var registEventID = (String)m_CaoController.Execute("RegistEventData",
registEventDataJson);

            // イベント情報の参照
            // ReferenceEventData
            paramList.Clear();
```

```
            paramList.Add(registEventID);
            var referenceEventData = m_CaoController.Execute("ReferenceEventData",
paramList.ToArray());

            // イベント情報の更新
            // UpdateEventData
            var updateEventDataJson = "{¥"event¥": {¥"notification¥": {¥"smtp¥": {¥"subject¥":
¥"温度が３１度超えました¥",¥"body¥": ¥"温度が３１度を超えています. 確認してください.
¥",¥"send_to¥": ¥"hoge@piyo.fujitsu.com¥"}},¥"conditions¥": {¥"targets¥":
{¥"read_access_code¥": ¥"TemperatureRU¥",¥"operations¥":
[¥"create¥",¥"update¥"],¥"resource_path¥":
¥"sensors/temperature¥"},¥"notification_condition¥": {¥"body_conditions¥": {¥"path¥":
¥"sensors/temperature¥",¥"path_type¥": ¥"JSONPath¥",¥"comparing_operator¥":
¥"ge¥",¥"value¥": 31.0}}}}}";
            paramStrList.Clear();
            paramStrList.Add(registEventID);
            paramStrList.Add(updateEventDataJson);
            m_CaoController.Execute("UpdateEventData", paramStrList.ToArray());

            // イベントの削除
            // DeleteEventData
            m_CaoController.Execute("DeleteEventData", registEventID);
        }
    }
}
```