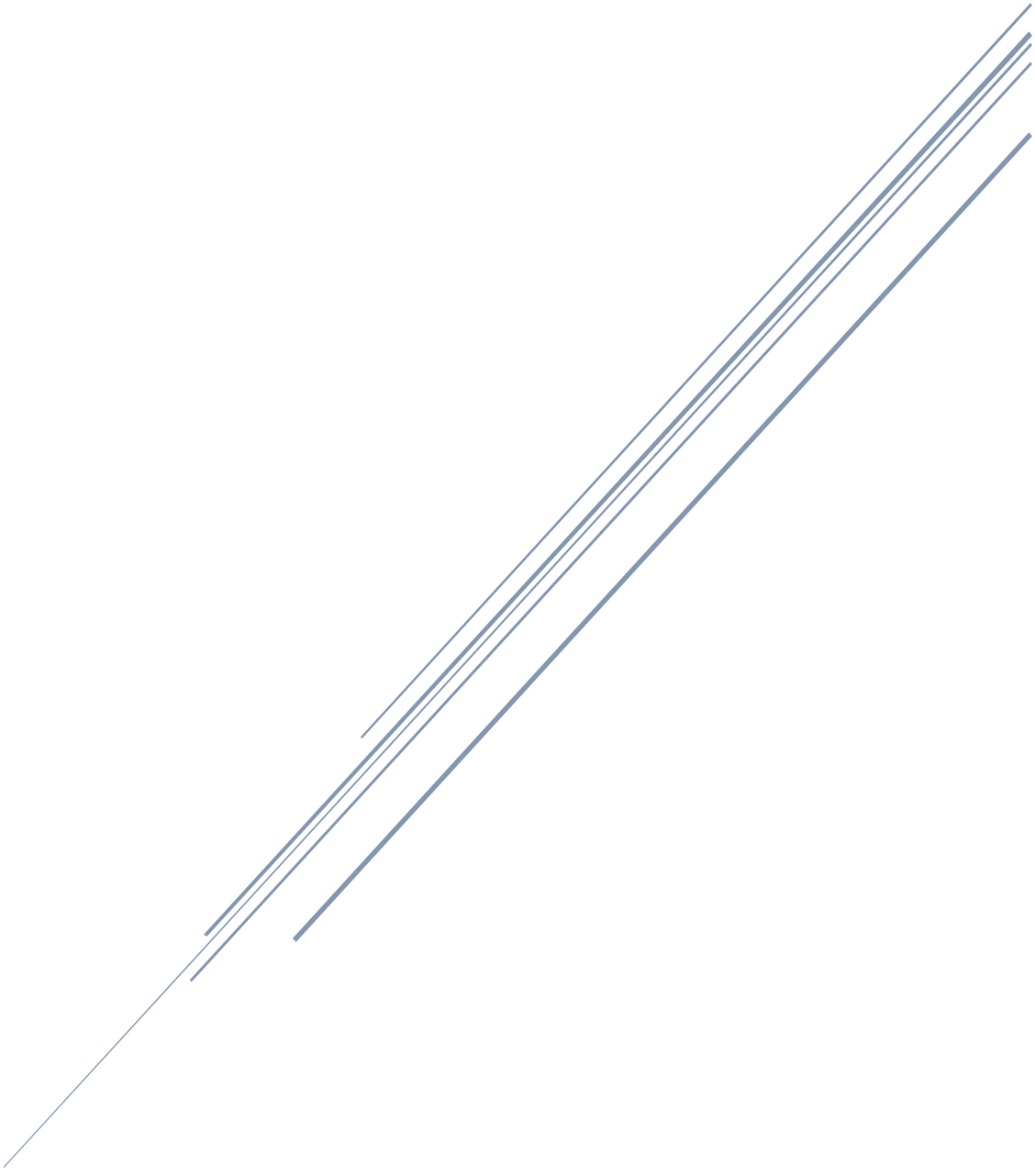
CIOF システム SDK 利用手順書

2022 年 1 月 20 日　Ver1.01



目次

1. 本書の目的

本書は、CIOF の提供する SDK を用いた実装方法を、例を用いて説明します。

2. 前提条件

本書のサンプル実装については、以下の環境を前提としております。

OS

Visual Studio のシステム要件に準拠します。

・Windows 10 バージョン 1703 以降:Home、Professional、Education、および Enterprise（LTSC および S はサポートされていません）

・Windows Server 2019:Standard および Datacenter

・Windows Server 2016:Standard および Datacenter

・Windows 8.1 (更新プログラム 2919355 を適用):Core、Professional、および Enterprise

・Windows Server 2012 R2（更新プログラム 2919355 を適用):Essentials、Standard、Datacenter

・Windows 7 SP1（最新の Windows Update を適用):Home Premium、Professional、Enterprise、Ultimate

開発環境

・Visual Studio 2015 以降

フレームワーク

・.Net Framework 4.6.2 以降

（開発用には developer pack が必要です。）

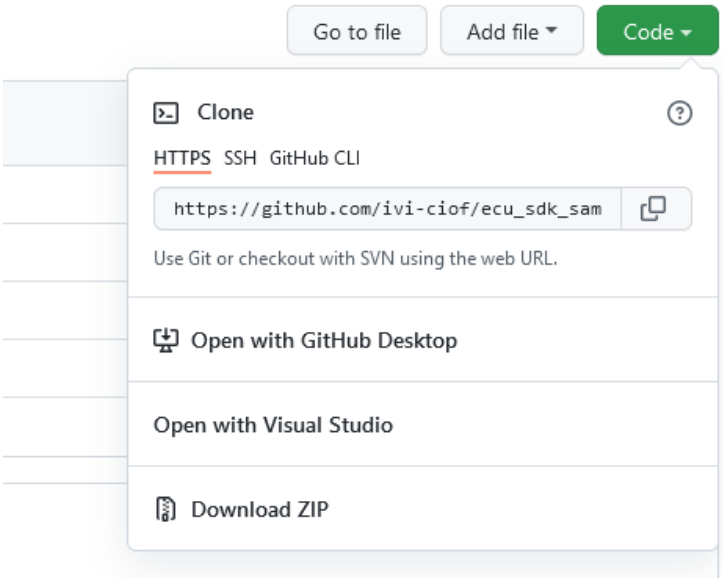
3. 配布形式

SDK とサンプルは、Nuget と Github を使って配布致します。

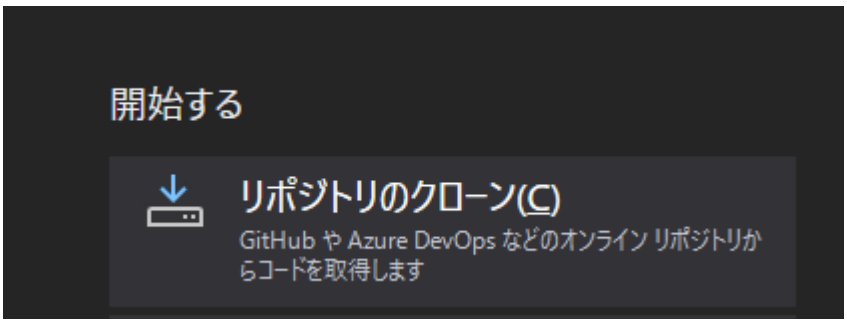
3.1. サンプルの取得

SDK サンプルは、以下の Github から入手できます。
https://github.com/ivi-ciof/ecu_sdk_sample

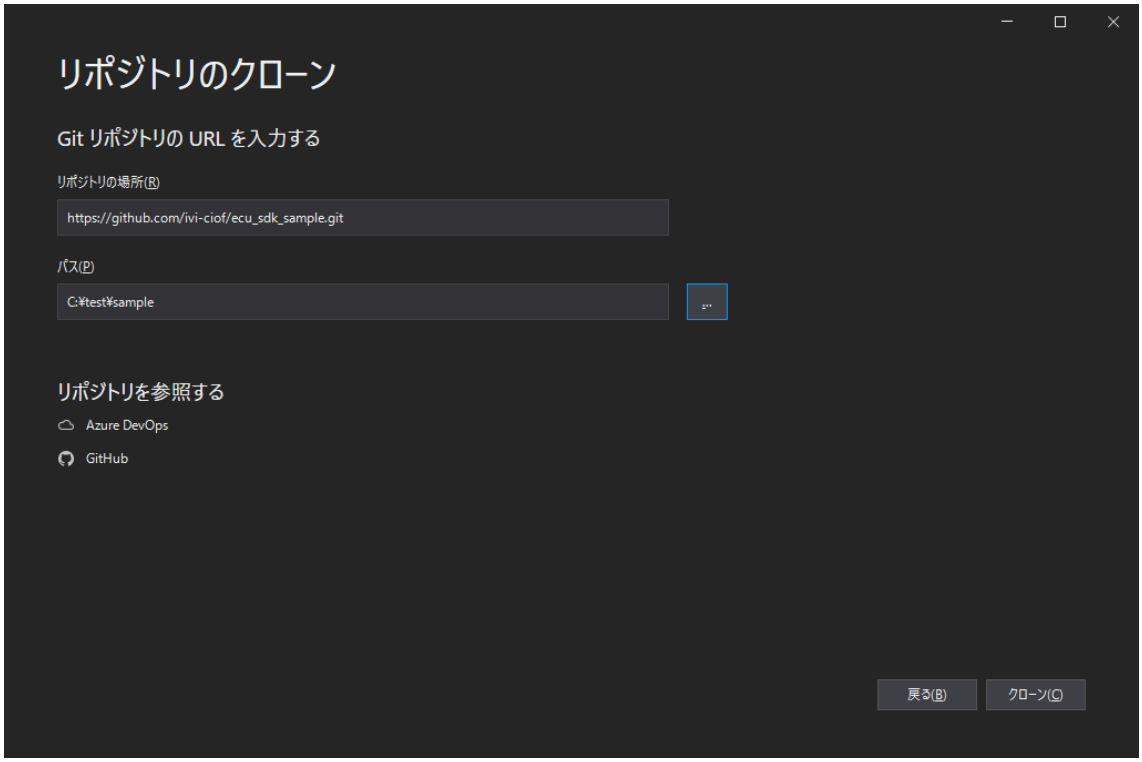
Code タブを選択して、Clone 用の URL をコピーします。



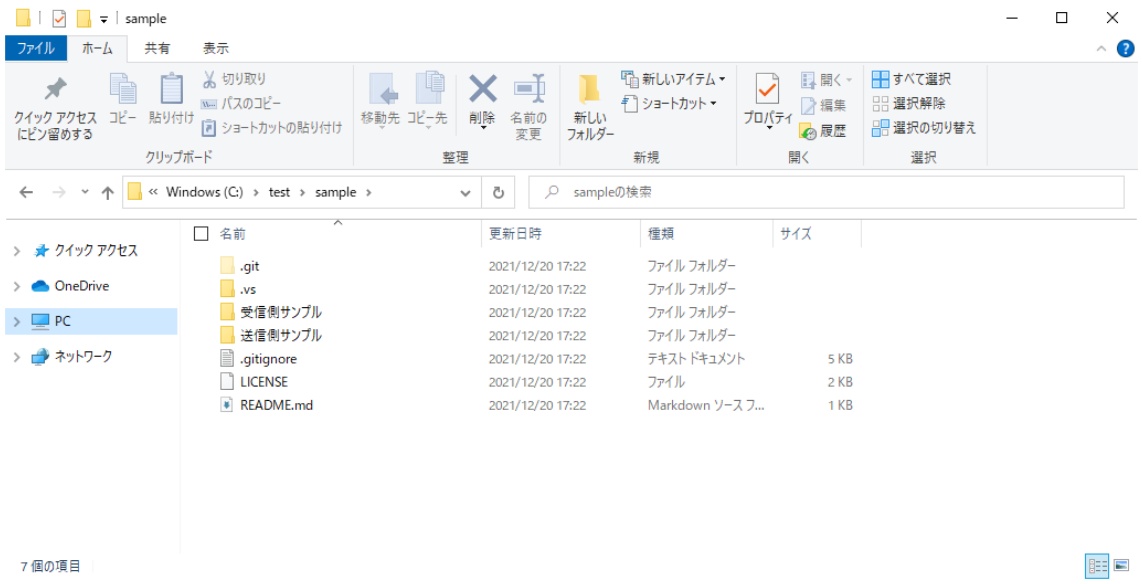
の Git 管理ツールを開きます。（ここでは VS2019 を例にして説明します。）



Github のパスとクローン先のフォルダを指定し、クローンを実行します。

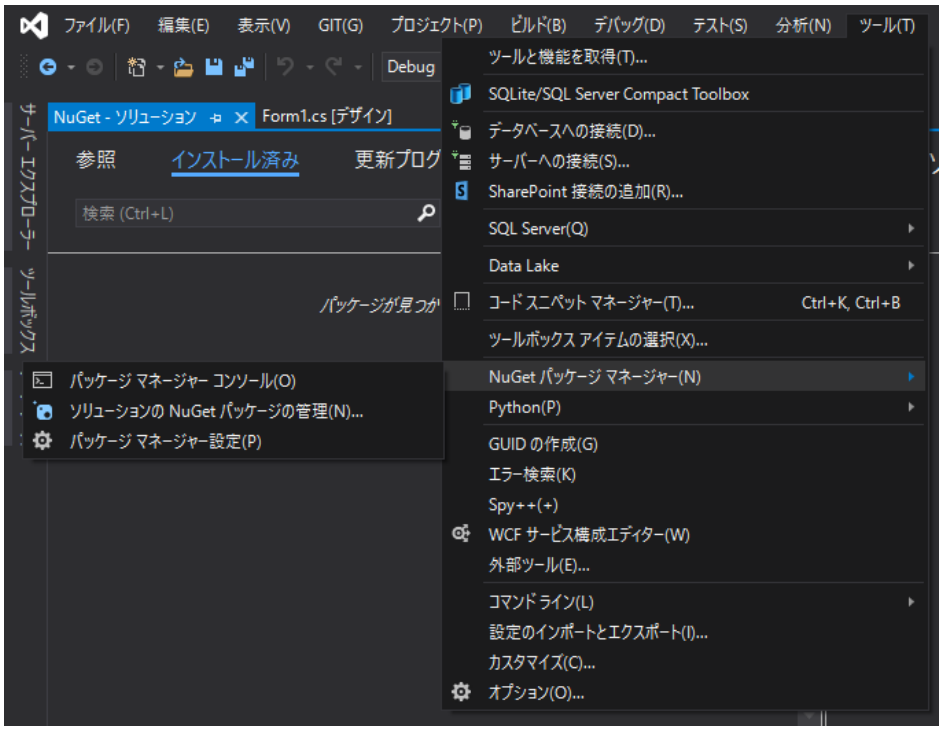


エクスプローラからソースがクローン出来ていることを確認します。



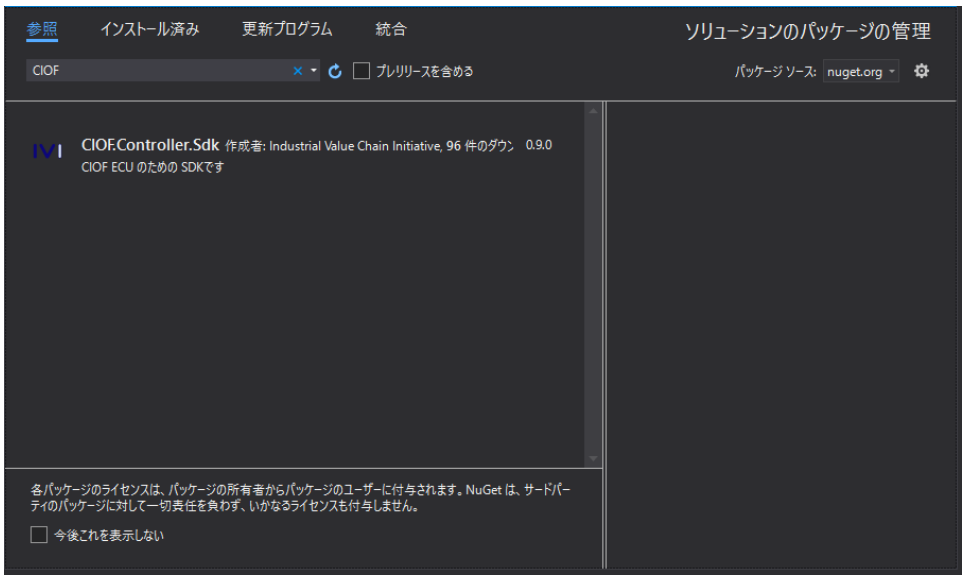
3.2. SDK の取得

Visual Studio の上部メニューからツール>NuGet パッケージマネージャー>ソリューションの NuGet パッケージの管理を選択します。

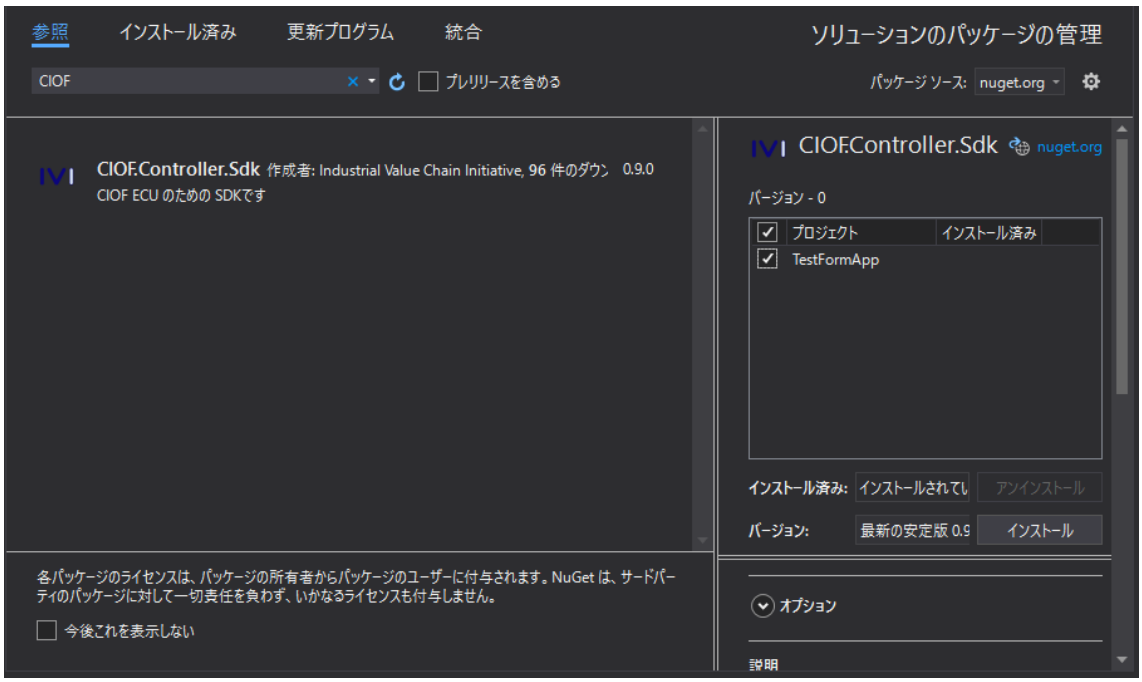


検索テキストボックスに、CIOF と入力して検索を行います。

CIOF.Controller.SDK を選択します。



インストール対象のプロジェクトをチェックして、インストールボタンをクリックします。



4. SDK クラス、メソッド一覧

ここでは、SDK のクラス、メソッド一覧を説明します。

4.1. SDK 全体クラス一覧

表 4.1-1 クラス名一覧

クラス名	説明
Namespace CIOF_SDK	
CIOFException	独自エラークラス
ControllerModel	コントローラの機能 Util クラス
ControllerRestUtil	連携サーバーへの REST API Util クラス
ControllerStatusConst	コントローラステータス定数
DataStatusConst	データ実装のステータス定数
EventTypeConst	イベントタイプ定数
FormatConst	フォーマットに関する定数
OutputLogger	Logger の Util クラス
RequestTypeConst	リクエストタイプ定数
ServiceStatusConst	サービス・プロセス・イベント実装のステータス定数
StatusCodeConst	ステータスコード定数
Namespace CIOF_SDK.Model	
Calendar	カレンダーモデル
CalendarRoot	カレンダー用のルートオブジェクト
CIOFError	CIOF 独自のエラー用モデル
Contract	取引契約モデル
ContractParameters	契約事項の契約項目
ContractRoot	取引契約用のルートオブジェクト
ControllerDataImplementation	データ実装
ControllerDataImplementationsRoot	コントローラ内のデータ実装定義
ControllerDataProperty	データ項目実装
ControllerEventImplementation	イベント実装
ControllerProcessImplementation	プロセス実装
ControllerServiceImplementation	サービス実装
ControllerServiceImplementationsRoot	コントローラ内のサービス実装定義

クラス名	説明
ControllerStatus	コントローラの状態通知用オブジェクト
DataImplementation	データ実装
DataImplementationsRoot	データ実装の RootObject
DataProperty	データ項目実装
DataPropertyStatus	データ項目実装(ステータス用)
DataStatus	データ実装(ステータス用)
DataStatusRoot	データ実装ステータス用のルート Obj
EdgeControllerAPIKey	コントローラ認証用 APIKey 受け渡し用モデル
EventImplementation	イベント実装
EventStatus	イベント実装ステータス
ProcessImplementation	プロセス実装
ProcessOperationImplementation	プロセス手順実装
ProcessStatus	プロセス実装ステータス
RequestParameter	リクエストパラメータ
RequestParemeterRoot	リクエストパラメータの Root オブジェクト
ServiceImplementation	サービス実装
ServiceImplementationsRoot	サービス実装定義のルートオブジェクト
ServiceRegister	サービス通知用オブジェクト
ServiceStatus	サービスステータス
ServiceStatusRoot	サービス実装状態通知の RootObj
TradeData	取引データ
TradeDataRoot	取引データのルートオブジェクト
Namespace CIOF_SDK.Util	
FileUtil	ファイル Util クラス
ServiceUtil	サービス実装用の Util クラス
TypeUtil	ユーザーに必要な型を提供する Util

4.2. 基本クラス

4.2.1. ControllerModel

コントローラの機能 Util クラス

表 4.2.1-1 プロパティ一覧

プロパティ名	説明
CalendarRoot	カレンダー情報
ContractRoot	取引契約情報
ControllerDataComponent	コントローラのデータ構造
ControllerServiceComponent	コントローラのサービス構造
ControllerStatus	コントローラのステータス
DataImplementationsRoot	データ実装情報
PollingRate	ポーリング周期(秒)
RequestParameterRoot	リクエストパラメータ
ServiceImplementationsRoot	サービス実装情報
TradeDataRoot	取引データ情報

表 4.2.1-2 メソッド一覧

メソッド名	説明
Close	終了処理
ExecuteCalendarEvent	カレンダーイベント実行
GetDataImplementations	データ実装状態取得
GetServiceImplementations	サービス実装状態取得
GetTradeData	取引データ情報取得
Init	初期化処理
InitialSetting	初期設定
IsControllerStart	コントローラの起動チェック
Polling	ポーリング処理
PollingStart	ポーリング開始処理
PostRequestParameter	リクエストパラメータの送信

メソッド名	説明
PostRequestParameterByRecordId	レコードIDを使用したリクエストパラメータの送信
PostRequestParameterByServiceId	サービスIDを使用したリクエストパラメータ送信
PostRequestParameterByServiceIdAndRecordId	サービスIDを使用した削除リクエストパラメータ送信
PostServiceRecord	サービス記録の通知
PostServiceRecordByDataId	取引データIDによるサービス記録の通知
PostServiceRecordByDataIdAndProcessIdAndEventId	取引データ ID およびプロセス ID とイベント ID によるサービス記録の通知
PostServiceRecordByProcessIdAndEventId	プロセス ID とイベント ID によるサービス記録の通知
PostTradeData	取引データの送信
PostTradeDataByServiceId	サービスIDを使用した取引データの送信
PutDataImplementationStatus	データ実装状態通知
PutDataImplementationStatusList	データ実装状態リスト通知
PutServiceImplementationsStatus	サービス実装状態通知
PutServiceImplementationStatusList	サービス実装状態通知
RestartCalendarEvent	カレンダーイベント再起動
SetDataMethod	データ ID の取得、保存に関するメソッド設定
SetProcessMethod	
SetServiceMethod	サービス実装メソッド設定
StartCalendar	カレンダーイベント開始
Stop	停止処理

4.3. Util クラス

4.3.1. ControllerRestUtil

連携サーバーへの REST API Util クラス

表 4.3.1-1 メソッド一覧

メソッド名	説明
Dispose	破棄処理
GetCalendar	カレンダー情報取得
GetContract	契約情報取得
GetDataImpementations	データ実装の取得
GetRequestParameter	リクエストパラメータの取得
GetServiceComponet	サービス実装取得
GetTradeData	取引データの取得
IsDispose	破棄されているか
PostRequestParameter	リクエストパラメータの送信
PostServiceRecord	サービス記録の通知
PostTradeData	取引データの送信
PutControllerStatus	コントローラの状態通知
PutDataImplemetationsStatus	データ実装の状態通知
PutServiceImplementations	サービス実装の状態設定

4.3.2. FileUtil

ファイル Util クラス

表 4.3.2-1 メソッド一覧

メソッド名	説明
GetAllFilePathList	特定のフォルダ下のファイルパスをすべて取得
GetEdgeControllerAPIKey	エッジコントローラ認証用 Yaml 読み込み処理
ReadJsonFile	Json ファイル読み込み
SaveJsonFile	Json ファイル保存

4.3.3. ServiceUtil

サービス実装用の Util クラス

表 4.3.3-1 メソッド一覧

メソッド名	説明
ConvertListToArray	任意のリストを配列に変換
CreateSendContractData	受信データから送信用の取引データを作成
GetContentList	送信内容リストの取得

4.3.4. TypeUtil

ユーザーに必要な型を提供する Util

表 4.3.4-1 メソッド一覧

メソッド名	説明
CreateSendData	string, object 型の Dictionary を返す。
CreateSendDataList	string, object 型の Dictiorany のリストを返す。

4.4. 定数クラス

4.4.1. ControllerStatusRoot

表 4.4.1-1 コントローラステータス定数

定数名	説明
DISCONNECTED	未接続
READY	稼働中
STOPPED	停止中

4.4.2. DataStatusConst

表 4.4.2-1 データ実装のステータス定数

定数名	説明
ABNORMAL	異常
EXCEPTION	例外
NORMAL	正常

4.4.3. EventTypeConst

表 4.4.3-1 イベントタイプ定数

定数名	説明
DELETE	削除
DUPLICATE	複製
READ	読取
REVISE	改変
STORE	保存
USE	利用

4.4.4. FormatConst

表 4.4.4-1 フォーマットに関する定数

定数名	説明
DATETIME_FORMAT	Datetime のフォーマット

4.4.5. RequestTypeConst

表 4.4.5-1 リクエストタイプ定数

定数名	説明
CREATE	生成
DELETE	削除

4.4.6. ServiceStatusConst

表 4.4.6-1 サービス・プロセス・イベント実装のステータス定数

定数名	説明
EXCEPTION	例外
FAILURE	故障
NORMAL	正常
STOP	停止

4.4.7. StatusCodeConst

表 4.4.7-1 ステータスコード定数

定数名	説明
BAD_REQUEST	400
SERVER_ERROR	500
SUCCESS	200
UNAUTHORIZED	401
UNPROCESSABLE_ENTITY	422

4.5. モデルクラス

4.5.1. Calender

カレンダー

表 4.5.1-1 プロパティ一覧

プロパティ名	説明
days_of_week	該当する曜日の配列
end_date	イベントの監視を終了する日時
id	定義したカレンダを識別する ID
interval	間隔に対応する数値
interval_type	間隔を示します。分"minute"，時間"hour"，日"day"，週"week"，月"month"，年"year"の区分
name	カレンダの実装名称
number_of_occrrences	イベントを実行する回数。この回数おこなったら終了する
recurrence_time_zone	該当するタイムゾーン
start_time	カレンダイベントを実行するにあたっての基準となる日時を GMT で示す
weeks_of_month	該当する週(1,2,3,4,5)の配列

4.5.2. CalenderRoot

カレンダー用のルートオブジェクト

表 4.5.2-1 プロパティ一覧

プロパティ名	説明
calendars	カレンダーの配列

4.5.3. CIOFError

CIOF 独自のエラー用モデル

表 4.5.3-1 プロパティ一覧

プロパティ名	説明
detail	エラー内容詳細
status	ステータスコード
title	エラー内容

4.5.4. Contract

取引契約

表 4.5.4-1 プロパティ一覧

プロパティ名	説明
companion_terminal_id	取引相手となるサイト ID
contract_parameters	契約事項の契約項目配列
contract_type	契約タイプ
data_implementation_id	対応するデータ実装 ID
data_implementation_local_id	対応するデータ実装の内部 ID
end_datetime	契約が終了した日時、終了前の場合は指定しない
event_implementations_id	イベント実装 ID リスト
event_implementations_local_id	イベント実装の内部 ID のリスト
id	取引契約の識別用 ID
process_implementation_id	プロセス実装 ID
process_implementation_local_id	プロセス実装の内部 ID
send_type	PUSH 型か PULL 型か
service_implementation_id	対応するサービス実装 ID
service_implementation_local_id	サービス実装の内部 ID
start_datetime	契約が成立した日時、開始前の場合は指定しない

4.5.5. ContractParameters

契約事項の契約項目

表 4.5.5-1 プロパティ一覧

プロパティ名	説明
content	契約事項の契約項目値
title	契約事項の契約項目名

4.5.6. ContractRoot

取引契約用のルートオブジェクト

表 4.5.6-1 プロパティ一覧

プロパティ名	説明
contracts	取引契約の配列

4.5.7. ControllerDataImplementation

データ実装

表 4.5.7-1 プロパティ一覧

プロパティ名	説明
data_property_implementations	データ項目実装配列
Id	ID
local_id	内部 ID

4.5.8. ControllerDataImplementationsRoot

コントローラ内のデータ実装定義

表 4.5.8-1 プロパティ一覧

プロパティ名	説明
data_implementations	データ実装の配列

4.5.9. ControlerDataProperty

データ項目実装

表 4.5.9-1 プロパティ一覧

プロパティ名	説明
id	ID
local_id	内部 ID

4.5.10. ControllerEventImplementation

イベント実装

表 4.5.10-1 プロパティ一覧

プロパティ名	説明
id	ID
local_id	内部 ID

4.5.11. ControllerProcessImplementation

プロセス実装

表 4.5.11-1 プロパティ一覧

プロパティ名	説明
event_implementations	イベント実装の配列
id	ID
local_id	内部 ID

4.5.12. ControllerServiceImplementation

サービス実装

表 4.5.12-1 プロパティ一覧

プロパティ名	説明
id	ID
local_id	内部 ID
process_implementations	プロセス実装の配列

4.5.13. ControllerServiceImplementationsRoot

コントローラ内のサービス実装定義

表 4.5.13-1 プロパティ一覧

プロパティ名	説明
service_implementations	サービス実装の配列

4.5.14. ControllerStatus

コントローラの状態通知用オブジェクト

表 4.5.14-1 プロパティ一覧

プロパティ名	説明
polling_rate	ポーリング周期
status	コントローラのステータス

4.5.15. DataImplementation

データ実装

表 4.5.15-1 プロパティ一覧

プロパティ名	説明
data_property_implementations	データ項目実装の配列
description	説明
id	ID
local_id	内部 ID
name	名前
service_implementation_id	データ実装を管理するサービス実装 ID

4.5.16. DataImplementationsRoot

データ実装の RootObject

表 4.5.16-1 プロパティ一覧

プロパティ名	説明
data_implementations	データ実装の配列

4.5.17. DataProperty

データ項目実装

表 4.5.17-1 プロパティ一覧

プロパティ名	説明
data_type	データ型
default_value	省略値
description	データ項目実装の簡単な説明
id	データ項目実装の ID
index	追番
is_primary_key	主キー
is_required	必須かどうか
local_id	データ項目実装の内部 ID
name	データ項目実装の名称

4.5.18. DataPropertyStatus

データ項目実装(ステータス用)

表 4.5.18-1 プロパティ一覧

プロパティ名	説明
id	説明
index	1 からの通し番号
name	データ項目実装の名称。該当する個別辞書の名称が対応する
remarks	通知内容
status	状態値

4.5.19. DataStatusRoot

データ実装ステータス用のルート Obj

表 4.5.19-1 プロパティ一覧

プロパティ名	説明
DataImplementationsStatuses	データ実装ステータスの配列

4.5.20. EdgeControllerAPIKey

コントローラ認証用 APIKey 受け渡し用モデル

表 4.5.20-1 プロパティ一覧

プロパティ名	説明
authorization_key	アクセスキー
id	コントローラ ID

4.5.21. EventImplementation

イベント実装

表 4.5.21-1 プロパティ一覧

プロパティ名	説明
description	イベント実装の説明
event_type	区分
id	イベント実装の ID
local_id	イベント実装の内部 ID
name	イベントの特徴を表す名称

4.5.22. EventStatus

イベント実装ステータス

表 4.5.22-1 プロパティ一覧

プロパティ名	説明
id	イベント実装の識別用 ID
local_id	イベント実装の内部 ID
remarks	備考
status	状態値

4.5.23. ProcessImplementation

プロセス実装

表 4.5.23-1 プロパティ一覧

プロパティ名	説明
description	プロセス実装の簡単な説明
event_implementations	イベント実装の配列
id	プロセス実装 ID
local_id	プロセス実装内部 ID
name	プロセス実装名称
process_operation_implementations	プロセス手順実装の配列

4.5.24. ProcessOperationImplementation

プロセス手順実装

表 4.5.24-1 プロパティ一覧

プロパティ名	説明
data_implementation_id	プロセスによって生成、利用、改変、または削除されるデータ実装 ID
id	プロセス手順実装の ID
index	1 からの通し番号
operation_type	区分

4.5.25. ProcessStatus

プロセス実装ステータス

表 4.5.25-1 プロパティ一覧

プロパティ名	説明
event_implementations	イベント実装ステータスの配列
id	プロセス実装の識別用 ID
local_id	プロセス実装の内部 ID
remarks	備考
status	状態値

4.5.26. RequestParameter

リクエストパラメータ

表 4.5.26-1 プロパティ一覧

プロパティ名	説明
condition	フィルタ条件
created_at	サービス実装が本リクエストパラメータを生成した生成日時
data_id	削除リクエストの場合に削除対象となる取引データ ID
domain_id	取引データを生成したドメイン ID
request_parameter_id	リクエストパラメータ ID
request_type	リクエスト区分
response_limit	本リクエストに対する応答期限
trade_contract_id	リクエストパラメータの対象となる取引契約 ID

4.5.27. ServiceInplementation

サービス実装

表 4.5.27-1 プロパティ一覧

プロパティ名	説明
description	サービス実装の簡単な説明
device_id	サービス実装が実際に実装されているデバイス ID の配列
id	サービス実装 ID
local_id	サービス実装の内部 ID
name	サービス実装の名称
process_implementations	プロセス実装の配列

4.5.28. ServiceImplementationsRoot

サービス実装定義のルートオブジェクト

表 4.5.28-1 プロパティ一覧

プロパティ名	説明
service_implementations	サービス実装の配列

4.5.29. ServiceRegister

サービス通知用オブジェクト

表 4.5.29-1 プロパティ一覧

プロパティ名	説明
data_id	取引データ ID
event_implementation_id	(任意)事実を記録したトリガとなったイベントの ID
event_implementation_local_id	(任意)事実を記録したトリガとなったイベントの内部 ID
event_type	(必須)事実区分
note	(任意)付記
result	(任意)結果値
service_implementation_id	(任意)事実を記録したサービス実装 ID
service_implementation_local_id	(任意)事実を記録したサービス実装の内部 ID
timestamp	(必須)記録日時

4.5.30. ServiceStatus

サービスステータス

表 4.5.30-1 プロパティ一覧

プロパティ名	説明
id	サービス実装の識別用 ID
local_id	サービス実装の内部 ID
process_implementations	プロセス実装ステータスの配列
remarks	通知内容
status	状態値

4.5.31. ServiceStatusRoot

サービス実装状態通知の RootObj

表 4.5.31-1 プロパティ一覧

プロパティ名	説明
service_status	サービスステータスの配列

4.5.32. TradeData

取引データ

表 4.5.32-1 プロパティ一覧

プロパティ名	説明
contents	データ項目内容
domain_id	ドメイン ID
headers	ヘッダ
id	取引データ ID
request_parameter_id	リクエストパラメータ ID
trade_contract_id	取引契約 ID

4.5.33. TradeDataRoot

取引データ のルートオブジェクト

表 4.5.33-1 プロパティ一覧

プロパティ名	説明
trade_datas	取引データの配列

4.6. Logger クラス

4.6.1. OutputLogger

Logger の Util クラス

表 4.6.1-1 メソッド一覧

メソッド名	説明
Dispose	終了時の処理
WriteDebugLog	Debug レベルの例外ログ出力
WriteEndTraceLog	Trace レベルの終了ログ出力
WriteErrorLog	Error レベルのログ出力
WriteFatalLog	Fatal レベルのログ出力
WriteInfoLog	Info レベルのログ出力
WriteStartTraceLog	Trace レベルの開始ログ出力
WriteTraceLog	Trace レベルのログ出力
WriteWarnLog	Warn レベルのログ出力

5. メソッド説明

ユーザーが使用する各メソッドを説明します。

5.1. ControllerModel クラスのメソッド

5.1.1. Close 終了処理

引数：なし

戻り値：なし

Close メソッド内では以下の処理を行っています。

・コントローラの状態通知

・コントローラの停止

5.1.2. ExecuteCalendarEvent カレンダーイベント実行

引数：

型	名称	説明
string	calendarId	カレンダーID

戻り値：なし

ExecuteCalendarEvent 内では以下の処理を行っています。

・カレンダー実装に基づいてイベントを実行

5.1.3. ExecuteTriggerEventProcess トリガイベントに紐づくプロセス実行

引数：

型	名称	説明
string	eventImplementationLocalId	イベント実装ローカル ID

戻り値：なし

ExecuteCalendarEvent 内では以下の処理を行っています。

・引数のイベント実装ローカル ID に紐づくプロセスを実行

5.1.4. GetCalendar カレンダー情報取得

引数：なし

戻り値：

型	説明
CalendarRoot	カレンダー情報

GetCalendar では以下の処理を行っています。

・API を使用してカレンダー情報を取得

5.1.5. GetContract 取引契約情報取得

引数：なし

戻り値：

型	説明
ContractRoot	取引情報

GetContract では以下の処理を行っています。

・API を使用して取引情報を取得

5.1.6. GetDataImplementations データ実装状態取得

引数：なし

戻り値：

型	説明
DataImplementationsRoot	データ実装状態

GetDataImplementation 内では以下の処理を行っています。

・API を使用してデータ実装状態を取得

5.1.7. GetServiceImplementations サービス実装状態取得

引数：なし

戻り値：

型	説明
ServiceImplementationsRoot	サービス実装状態

GetServiceImplementationsRoot 内では以下の処理を行っています。

・API を使用してサービス実装状態を取得

5.1.8. GetTradeData 取引データ情報取得

引数：

型	名称	説明
String	serviceImplementationId	サービス実装 ID（default 値：null）
String	serviceImplementationLocalId	サービス実装ローカル ID（default 値：null）
String	contractID	契約 ID（default 値：null）

戻り値：

型	説明
TradeDataRoot	取引データ情報

GetTradeData 内では以下の処理を行っています。

・API を使用して取引データ情報を取得

・取引データ情報取得した旨を通知

5.1.9. Init 初期化処理

引数：

型	名称	説明
String	path	認証ファイルパス

戻り値：なし

Init では以下の処理を行っています。

・認証ファイルパスの読み込み

※InitialSetting、PollingStart を呼び出す前に Init を呼び出してください。

5.1.10. InitilalSetting 初期設定

引数：

型	名称	説明
string	hostname	ホスト名
int	rate	ポーリング周期
string	serviceComponentFilePath	サービス実装構成ファイル
string	dataComponentFilePath	データ実装構成ファイル

戻り値：なし

InitialSetting メソッド内では以下の処理を行っています。

・コントローラとの通信用オブジェクトを作成。

これにより API 用メソッドが使用できます。

・ポーリング周期設定

・コントローラの状態通知

・サービス ID の内部リスト取得

・データ ID の内部リスト取得

・取引情報取得

・カレンダー情報取得

・サービスプロファイル取得

・データプロファイル取得

・カレンダー起動イベント設定

5.1.11. IsControllerStart コントローラの起動チェック

引数：なし

戻り値：なし

IsControllerStart メソッド内では以下の処理を行っています。

・コントローラの起動チェック

5.1.12. Polling ポーリング処理

引数：なし

戻り値：なし

Polling メソッド内では以下の処理を行っています。

・リクエストパラメータの取得

→リクエストを受信した場合、登録されたサービス実装メソッドを実行

・取引データ情報の取得

→取引データを受信した場合、登録されたサービス実装メソッドを実行

5.1.13. PollingStart ポーリング開始処理

引数：なし

戻り値：なし

PollingStart メソッド内では以下の処理を行っています。

・サービス実装の構成チェック→状態通知

・データ実装の構成チェック→状態通知

・コントローラの状態通知

・ポーリング開始

5.1.14. PostRequestParameter リクエストパラメータの送信

引数：

型	名称	説明
string	tradeContractId	取引契約 ID
string	requestType	リクエストの種類
string	condition	予め取引事業者間で定めたルールの記述（default 値：null）
string	tradeDataId	取引データ ID（default 値：null）
string	domainId	ドメイン ID（default 値：null）
Datetime（Nullable）	responseLimit	本リクエストに対する応答期限（default 値：null）

戻り値：なし

PostRequestParameter 内では以下の処理を行っています。

・API を使用してリクエストパラメータの送信

5.1.15. PostRequestParameterByRecordId レコード ID を使用したリクエストパラメータの送信

引数：

型	名称	説明
string	tradeContractId	取引契約 ID
string	requestType	リクエストの種類
string	condition	予め取引事業者間で定めたルールの記述（default 値：null）
string	recordId	レコード ID(デフォルト値: null)
string	domainId	ドメイン ID(デフォルト値: null)
Datetime（Nullable）	responseLimit	本リクエストに対する応答期限（default 値：null）

戻り値：なし

PostRequestParameterByRecordId メソッド内では以下の処理を行っています。

・レコード ID からデータ ID を取得

・リクエストパラメータの送信

5.1.16. PostRequestParameterByServiceId サービス ID を使用したリクエストパラメータ送信

引数：

型	名称	説明
string	serviceImplementationLocalId	サービス実装ローカル ID
string	requestType	リクエストの種類
string	condition	予め取引事業者間で定めたルールの記述（default 値：null）
string	tradeDataId	取引データ ID(デフォルト値：null)
string	domainId	ドメイン ID（デフォルト値：null）
Datetime（Nullable）	responseLimit	本リクエストに対する応答期限（default 値：null）

戻り値：なし

PostRequestParameterByServiceId では以下の処理を行っています。

・サービス ID から契約リストを取得

・リクエストパラメータを送信

5.1.17. PostRequestParameterByServiceIdAndRecordId サービスIDを使用した削除リクエストパラメータ送信

引数：

型	名称	説明
string	serviceImplementationLocalId	サービス実装ローカルID
string	requestType	リクエストの種類
string	condition	予め取引事業者間で定めたルールの記述（default値：null）
string	recordId	レコードID（デフォルト値：null）
string	domainId	ドメインID（デフォルト値：null）
Datetime（Nullable）	responseLimit	本リクエストに対する応答期限（default値：null）

戻り値：なし

PostRequestParameterByServiceAndRecordId メソッド内では以下の処理を行っています。

・レコードIDからデータIDを取得

・リクエストパラメータの送信

5.1.18. PostServiceRecord サービス記録の通知

引数：

型	名称	説明
string	recordId	レコード ID
string	eventType	イベントタイプ
string	note	事実に関する記述（default 値：null）
string	result	事実に関する記述（default 値：null）
string	serviceImplementationLocalId	サービス実装ローカル ID（default 値：null）
string	eventImplementationLocalId	イベント実装ローカル ID（default 値：null）

戻り値：なし

PostServiceRecord メソッド内では以下の処理を行っています。

・レコード ID からデータ ID を取得

・サービス記録の通知

5.1.19. PostServiceRecordByDataId 取引データ ID によるサービス記録の通知

引数：

型	名称	説明
string	tradeDataId	取引データ ID
string	eventType	イベントタイプ
string	note	事実に関する記述（default 値：null）
string	result	事実に関する値（default 値：null）
string	serviceImplementationLocalId	サービス実装ローカル ID（default 値：null）
string	eventImplementationLocalId	イベント実装ローカル ID（default 値：null）

戻り値：なし

PostServiceRecordByDataId では以下の処理を行っています。

・API を使用してサービス記録の通知

5.1.20. PostServiceRecordByDataIdAndProcessIdAndEventId 取引データおよびプロセス ID とイベント ID によるサービス記録の通知

引数：

型	名称	説明
string	tradeDataId	取引データ ID
string	eventType	イベントタイプ
string	note	事実に関する記述（default 値：null）
string	result	事実に関する値（default 値：null）
string	processImplementationLocalId	プロセス実装ローカル ID（default 値：null）
string	eventImplementationLocalId	イベント実装ローカル ID（default 値：null）

戻り値：なし

PostServiceRecordByDataIdAndProcessIdAndEventId では以下の処理を行っています。

・イベントのローカル ID からイベントを取得

・サービス記録の通知

5.1.21. PostServiceRecordByProcessIdAndEventId プロセス ID とイベント ID によるサービス記録の通知

引数：

型	名称	説明
string	recordId	レコード ID
string	eventType	イベントタイプ
string	note	事実に関する記述（default 値：null）
string	result	事実に関する値（default 値：null）
string	processImplementationLocalId	プロセス実装ローカル ID（default 値：null）
string	eventImplementationLocalId	イベント実装ローカル ID（default 値：null）

戻り値：なし

PostServiceRecordByProcessIdAndEventId では以下の処理を行っています。

・レコード ID からデータ ID を取得

・サービス記録の通知

5.1.22. PostTradeData 取引データの送信

引数：

型	名称	説明
string	tradeContractId	契約 ID
List<Dictionary<string,object>>	sendContentsList	送信するデータの名称と値の Map(key：個別辞書で定義したデータの名称, value：送信する実データ)リスト
string	tradeDataId	取引データ ID（default 値：null）
string	requestParameterId	リクエストパラメータ ID（default 値：null）
string	domainId	ドメイン ID (default 値：null)

戻り値：

型	説明
string	取引データ ID（サーバで付与される）

PostTradeData では以下の処理を行っています。

・送信内容の設定

・API を使用して取引データの送信

・サーバで付与された取引データ ID を取得

5.1.23. PostTradeDataByServiceId サービス ID を使用した取引データの送信

引数：

型	名称	説明
string	serviceImplementationLocalId	サービス実装ローカル ID
List<Dictionary<string,object>>	sendContentsList	送信するデータの名称と値の Map(key：個別辞書で定義したデータの名称, value：送信する実データ)リスト
string	tradeDataId	取引データ ID（default 値：null）
string	requestParameterId	リクエストパラメータ ID(default 値：null)
string	domainId	ドメイン ID(default 値：null)

戻り値：

型	説明
List<string>	データ ID リスト

PostTradeDataByServiceId では以下の処理を行っています。

・localServiceId から契約リストを取得

・契約ごとに取引データの送信

・データ ID のリストを返却

5.1.24. PutDataImplementationStatus データ実装状態通知

引数：

型	名称	説明
DataStatus	dataStatus	データ実装状態
List<DataPropertyStatus>	dataPropertyStatusList	データ項目実装状態リスト

戻り値：なし

PutDataImplementationStatus では以下の処理を行っています。

・DataStatus のリストを作成

・API を使用してデータ実装状態を通知

5.1.25. PutDataImplementationStatusList データ実装状態リスト通知

引数：

型	名称	説明
List<DataStatus>	dataStatusList	データ実装状態リスト

戻り値：なし

PutDataImplementationStatusList では以下の処理を行っています。

・API を使用してデータ実装状態を通知

5.1.26. PutServiceImplementationsStatus サービス実装状態通知

引数：

型	名称	説明
ServiceStatus	serviceStatus	サービス実装状態
ProcessStatus	processStatus	プロセス実装状態
EventStatus	eventStatus	イベント実装状態

戻り値：なし

PutServiceImplementationsStatus では以下の処理を行っています。

・ServiceStatus のリストを作成

・API を使用してサービス実装状態を通知

5.1.27. PutServiceImplementationsStatusList サービス実装状態通知

引数：

型	名称	説明
List<ServiceStatus>	serviceStatusList	サービス実装状態リスト

戻り値：なし

PutServiceImplementationsStatusList では以下の処理を行っています。

・API を使用してサービス実装状態を通知

5.1.28. RestartCalendarEvent カレンダーイベント再起動

引数：なし

戻り値：なし

RestartCalendarEvent では以下の処理を行っています。

・カレンダー実装情報を再取得

・カレンダーイベント実行処理を再起動

5.1.29. SetDataMethod データ ID の取得、保存に関するメソッド設定

引数：

型	名称	説明
Action	saveDataId	データ ID 保存メソッド（引数なし、戻り値なし）
Func<string,string>	getDataId	データ ID 取得メソッド（引数：レコード ID、戻り値：データ ID）

戻り値：なし

SetDataMethod では以下の処理を行っています。

・データ ID 保存メソッドの設定

※データ ID 保存メソッドは以下のメソッド内で呼び出されます。

5.1.1 Close

・データ ID 取得メソッドの設定

※データ ID 取得メソッドは以下のメソッド内で呼び出されます。

5.1.13 PostRequestParameterByRecordId

5.1.15 PostRequestParameterByServiceIdAndRecordId

5.1.16 PostServiceRecord

5.1.19 PostServiceRecordByProcessIdAndEventId

5.1.30. SetProcessMethod プロセス実装メソッド設定

引数：

型	名称	説明
Dictionary<string,Action>	processDict	プロセス実装メソッド Map(key:プロセス実装ローカル ID：value:プロセス実装メソッド（引数なし）)

戻り値：なし

SetProcessMethod では以下の処理を行っています。

・プロセス実装メソッドの設定

※設定したプロセス実装メソッドは以下のメソッド内で呼び出されます。

5.1.2 ExecuteCalendarEvent

5.1.31. SetServiceMethod サービス実装メソッド設定

引数：

型	名称	説明
Dictionary<string, Action<string, Dictionary<string, List<Dictionary<string, object>>>>>	serviceImplementsMethodDictList	サービス実装メソッド Map(key:サービス実装ローカル ID：value:サービス実装メソッド（引数 contractId，取得データ Map(key：name，value：content）のリスト))
Dictionary<string, Action<string, string, string>>	createRequestServiceImplementsMethodListDict	生成リクエストパラメータ用サービス実装メソッド Map(key:サービス実装ローカル ID：value:サービス実装メソッド（引数 condition, response_limit, trade_contract_id))
Dictionary<string, Action<string, string, string, string>>	deleteRequestServiceImplementsMethodListDict	削除リクエストパラメータ用サービス実装メソッド Map(key:サービス実装ローカル ID：value:サービス実装メソッド（引数 condition, response_limit,

型	名称	説明
		data_id, trade_contract_id)

戻り値：なし

SetServiceMethod では以下の処理を行っています。

・サービス実装メソッドの設定

※設定したサービス実装メソッドは以下のメソッド内で呼び出されます。

5.1.10 Polling

・生成リクエストパラメータ用サービス実装メソッドの設定

※設定した生成リクエストパラメータ用サービス実装メソッドは以下のメソッド内で呼び出されます。

5.1.10 Polling

・削除リクエストパラメータ用サービス実装メソッドの設定

※設定した削除リクエストパラメータ用サービス実装メソッドは以下のメソッド内で呼び出されます。

5.1.10 Polling

5.1.32. StartCalendar カレンダーイベント開始

引数：なし

戻り値：なし

StartCalendar では以下の処理を行っています。

・カレンダーイベントの実行

5.1.33. Stop 停止処理

引数：なし

戻り値：なし

Stop では以下の処理を行っています。

・コントローラ状態を通知

・コントローラの停止

5.2. ControllerRestUtil クラスのメソッド

5.2.1. Dispose 破棄処理

引数：なし

戻り値：なし

Dispose では以下の処理を行っています。

・ControllerRestUtil オブジェクトの破棄

5.2.2. GetCalendar カレンダー情報取得

引数：なし

戻り値：

型	説明
CalendarRoot	カレンダー情報の RootObject

GetCalendar では以下の処理を行っています。

・以下の API を使用してカレンダー情報取得

HTTP メソッド：GET

URL パス：/hct/api/v2/calenders

5.2.3. GetContract 契約情報取得

引数：なし

戻り値：

型	説明
ContractRoot	契約情報の RootObject

GetContract では以下の処理を行っています。

・以下の API を使用して契約情報取得

HTTP メソッド：GET

URL パス：/hct/api/v2/trade_contracts

5.2.4. GetDataImplementatons データ実装の取得

引数：なし

戻り値：

型	説明
DataImplementationsRoot	データ実装の RootObject

GetDataImplementations では以下の処理を行っています。

・以下の API を使用してデータ実装を取得

HTTP メソッド：GET

URL パス：/hct/api/v2/data_implementations

5.2.5. GetRequestParameter リクエストパラメータの取得

引数：なし

戻り値：

型	説明
RequestParameterRoot	リクエストパラメータ RootObject

GetRequestParameter では以下の処理を行っています。

・以下の API を使用してデータ実装を取得

HTTP メソッド：GET

URL パス：/hct/api/v2/requests

5.2.6. GetServiceComponent サービス実装取得

引数：なし

戻り値：

型	説明
ServiceImplementationsRoot	サービス実装の RootObject

GetServiceComponent では以下の処理を行っています。

・以下の API を使用してサービス実装を取得

HTTP メソッド：GET

URL パス：/hct/api/v2/ service_implementations

5.2.7. GetTradeData 取引データの取得

引数：

型	名称	説明
string	serviceImplementationId	サービス実装 ID（default 値：null）
string	serviceImplementationLocalId	サービス実装ローカル ID（default 値：null）
string	tradeContractId	取引契約 ID（default 値：null）

GetTradeData では以下の処理を行っています。

・以下の API を使用してサービス実装を取得

HTTP メソッド：GET

URL パス：

/hct/api/v2/messages?&service_implementation_id=< serviceImplementationId >

&service_implementation_local_id=< serviceImplementationLocalId >

&trade_contract_id=< tradeContractId >

5.2.8. IsDispose 破棄されているか

引数：なし

戻り値：

型	説明
bool	破棄されている場合は true

IsDispose では以下の処理を行っています。

・HTTPClient オブジェクトが破棄されているか確認。

破棄されている場合は true を返す。

5.2.9. PostRequestParameter リクエストパラメータの送信

引数：

型	名称	説明
RequestParameterRoot	requestParameterRoot	リクエストパラメータの RootObject

戻り値：なし

PostRequestParameter では以下の処理を行っています。

・以下の API を使用してリクエストパラメータを送信

HTTP メソッド：POST

リクエスト URL：/hct/api/v2/requests

5.2.10. PostServiceRecord サービス記録の通知

引数：

型	名称	説明
ServiceRegister	requestData	サービス記録の Object

戻り値：なし

PostServiceRecord では以下の処理を行っています。

・以下の API を使用してリクエストパラメータを送信

HTTP メソッド：POST

リクエスト URL：/hct/api/v2/service_record

5.2.11. PostTradeData 取引データの送信

引数：

型	名称	説明
TradeDataRoot	contractDataRoot	取引データの RootObject

戻り値：なし

PostTradeData では以下の処理を行っています。

・以下の API を使用してリクエストパラメータを送信

HTTP メソッド：POST

リクエスト URL：/hct/api/v2/messages

5.2.12. PutControllerStatus コントローラの状態通知

引数：

型	名称	説明
ControllerStatus	controllerStatus	コントローラ状態 object

戻り値：なし

PutControllerStatus では以下の処理を行っています。

・以下の API を使用してコントローラの状態を通知

HTTP メソッド：PUT

リクエスト URL：/hct/api/v2/controller

5.2.13. PutDataImplementationsStatus データ実装の状態通知

引数：

型	名称	説明
DataStatusRoot	dataStatusRoot	データ実装の状態通知用 RootObject

戻り値：なし

PutDataImplementationsStatus では以下の処理を行っています。

・以下の API を使用してコントローラの状態を通知

HTTP メソッド：PUT

リクエスト URL：/hct/api/v2/data_implementations

5.2.14. PutServiceImplementations サービス実装の状態設定

引数：

型	名称	説明
ServiceStatusRoot	serviceStatusRoot	サービス実装ステータスの RootObject

戻り値：なし

PutServiceImplementationsStatus では以下の処理を行っています。

・以下の API を使用してコントローラの状態を通知

HTTP メソッド：PUT

リクエスト URL：/hct/api/v2/service_implementations

5.3. FileUtil のメソッド

5.3.1. GetAllFilePathList 特定のフォルダ下のファイルパスをすべて取得

引数：

型	名称	説明
string	folderPath	フォルダパス

戻り値：

型	説明
List<string>	ファイルパスのリスト

GetAllFilePathList では以下の処理を行っています。

・引数で指定されたフォルダ内のファイルパスをすべて取得

5.3.2. GetEdgeControllerAPIKey エッジコントローラ認証用 yml 読み込み処理

引数：

型	名称	説明
string	path	yml ファイルのパス

戻り値：

型	説明
EdgeControllerAPIKey	コントローラの認証情報 Object

GetEdgeControllerAPIKey では以下の処理を行っています。

・引数で指定された yml ファイルを読み込みコントローラの認証情報を返却

5.3.3. ReadJsonFile<T> JSON ファイル読み込み

引数：

型	名称	説明
string	path	ファイルパス

戻り値：

型	説明
T（メソッド呼び出し時に指定）	json をデシリアライズしたオブジェクト

ReadJsonFile では以下の処理を行っています。

・引数で指定された json ファイルを読み込みデシリアライズして返却

5.3.4. SaveJsonFileBySpecifiedFileName<T> 指定のファイル名で JSON ファイル保存

引数：

型	名称	説明
T（メソッド呼び出し時に指定）	modelObject	モデル用オブジェクト
string	filePath	保存先ファイルパス

戻り値：なし

SaveJsonFileBySpecifiedFileName では以下の処理を行っています。

・引数で指定されたファイル名で Json ファイルを保存

5.3.5. SaveJsonFileToSpecifiedFolder<T>

引数：

型	名称	説明
T（メソッド呼び出し時に指定）	modelObject	モデル用オブジェクト
string	destFolderPath	保存先フォルダパス
string	fileName	ファイル名(default 値:null)

戻り値：なし

SaveJsonFileToSpecifiedFolder では以下の処理を行っています。

・引数で指定されたフォルダに json ファイルを保存

※引数のファイル名が null の場合、yyyyMMddHHmmss.json で保存されます。

5.3.6. ShowSaveFileDialog ファイル保存ダイアログ表示

引数：

型	名称	説明
string	path	ファイル保存パス
string	filter	フィルタ

戻り値：なし

ShowSaveFileDialog では以下の処理を行っています。

・ファイル保存用のダイアログを表示

5.4. ServiceUtil クラスのメソッド

5.4.1. ConvertListToArray<T> 任意のリストを配列に変換

引数：

型	名称	説明
List<T（メソッド呼び出し時に指定）>	modelList	リスト

戻り値：

型	説明
T[]	配列

ConvertListToArray では以下の処理を行っています。

・引数で渡されたリストを配列に変換して返却

5.4.2. CreateSendContractData 受信データから送信用の取引データを作成

引数：

型	名称	説明
string	contractId	契約 ID
object[]	dataArray	データ配列
RequestParameter	requestParameter	リクエストパラメータ

戻り値：

型	説明
TradeDataRoot	取引データルートオブジェクト

CreateSendContractData では以下の処理を行っています。

・引数の契約 ID とデータ配列から取引データオブジェクトを作成し返却

5.4.3. GetContentList 送信内容リストの取得

引数：

型	名称	説明
string	contractId	契約 ID
ContractRoot	dataArray	契約情報
DataImplementationsRoot	dataImplementationsRoot	データ実装ルートオブジェクト
List<Dictionary<string, object>>	contentDictList	送信内容

戻り値：

型	説明
List<object[]>	送信内容リスト

CreateSendContractData では以下の処理を行っています。

・引数の送信内容とデータ実装定義から送信内容リストを作成し返却

5.5. TypeUtil のメソッド

5.5.1. CreateSendData

引数：なし

戻り値：

型	説明
Dictionary<stirng,object>	送信データ用オブジェクトの Map

CreateSendData では以下の処理を行っています。

・送信データ用オブジェクトの Map を返却

5.5.2. CreateSendDataList

引数：なし

戻り値：

型	説明
List<Dictionary<stirng,object>>	送信データ用オブジェクトの Map の List

CreateSendData では以下の処理を行っています。

・送信データ用オブジェクトの Map のリストを返却

6. SDK のメソッド呼び出し順序想定例

4. SDK クラス、メソッド一覧 で紹介した SDK のメソッドの使用方法について、
以下のような流れを想定しています。
※目的によって、呼び出す処理や順序は変わります。

１．連携ターミナルとの通信用の認証ファイル読込の為 ControllerModel.Init を呼び出す。
このメソッド内で、連携ターミナルと通信するための認証ファイルを読み込みます。

２．初期化処理として、ControllerModel.InitialSetting を呼び出す。
このメソッド内で、コントローラとの通信用オブジェクトの生成、
ポーリング周期の設定、コントローラの状態通知、
サービス実装内部 ID、データ実装内部 ID のリスト取得を行います。

３．サービス実装のメソッド登録の為に ControllerModel.SetServiceMethod を呼び出す。
このメソッドの引数に、サービス実装内部 ID とサービス実装メソッドの Map
のリストを渡して、コントローラにコールバックで呼び出すメソッドを登録します。

このメソッドの引数には、下記を指定し、コントローラにサービス実装としてコールバックで呼び出すメソッドを登録します。
サービス実装内部 ID とサービス実装メソッドの Map の リスト

サービス実装メソッドの引数には以下を指定します。
- 第一引数：契約 ID(型は string)
- 第二引数：送信されるデータリスト（型は List<Dictionary<string, object>>）

コールバックメソッドの想定される例
・コントローラがデータ送信側の場合の想定；データ送信メソッド
ControllerModel の PostTradeData メソッドを使用して、
受信側に取引データを送信する。
このメソッドの引数として下記を指定する。
- 第一引数：契約 ID（コールバックメソッドの第一引数に入ってきた値をそのまま渡す）
- 第二引数:送信データリスト(データ実装で定義されている名称を Key として、実データを Value とした Map のリスト)

・コントローラがデータ受信側の場合の想定：データ保管メソッド

コールバックメソッドの第一引数に契約 ID、第二引数に受信したデータが入ってきます。

第一引数に入ってくる契約 ID と、ControllerModel の trade_contract_id が一致する取引データを取得します。

取引データの ID と受信したデータを紐づけて内部に保管します。

※データを受信した際のサービス通知（event_type：読取）は
SDK で行っているので、改めて通知を行う必要はありません。

4．取引データ ID とローカルのデータ ID との紐づけ用コールバックメソッド登録の為に ControllerModel.SetDataMethod を呼び出す。

このメソッドの引数に下記を指定し

- 第一引数：データ ID とローカル ID の紐づけ保存用メソッド
- 第二引数：ローカル ID からデータ ID を取得するメソッド
- 第三引数：データ ID からローカル ID リストを取得するメソッド

コントローラにコールバックで呼び出すメソッドを登録します。

5．ポーリング開始の為に、ControllerModel.PollingStart メソッドを呼び出す。

このメソッドを呼び出すことで、ポーリング処理を開始します。

このメソッド内では、ポーリング開始前に下記を実行します。

- ControllerRestUtil の GetContract メソッドによる取引契約情報の取得
- ControllerRestUtil の GetCalender メソッドによるカレンダー情報取得

ポーリング処理内では下記の処理を実行しています。

- ControllerRestUtil の GetServiceComponet メソッドによる
  サービスプロファイル取得
- ControllerRestUtil の GetDataImpementations メソッドによる
  データプロファイル取得
- ControllerRestUtil の GetRequestParameter によるリクエストパラメータの取得
- ControllerModel の GetTradeData メソッドによる取引データの取得

取得した契約情報やリクエストパラメータに対応するサービスが
登録されている場合は、サービスを実行します。

※ポーリング内で GetTradeData による取引データの取得を行っているため、
コントローラ内で再度取引データの取得を行う必要はありません。
一度取得した取引データはサーバーから削除されるようになっております。

7. サンプルソース説明

SDK を使用した 2 つのサンプルについて説明します。
このサンプルは、連携マネージャで設定した取引契約に従って、
データの送受信を行うサンプルです。
送信側のサンプルが、CIOFContractSample_Factory
受信側のサンプルが、CIOFContractSample です。
送信側は Push 型の契約の場合のデータ送信と、
Pull 型の場合のリクエストパラメータに応じたデータ送信に対応しています。

7.1. サンプル動作環境設定

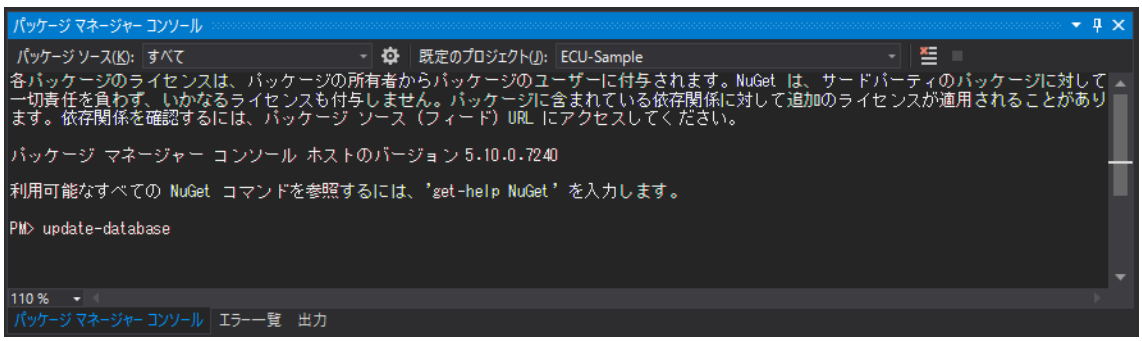
サンプルでは、送信・取得したデータを保存するために SQLite が必要です。
ここでは SQLite のインストールについて説明します。

以下のページから、sqlite-netFx46-setup-bundle-x64-2015-1.0.115.0.exe
をダウンロードします。
https://system.data.sqlite.org/index.html/doc/trunk/www/downloads-unsup.wiki

ダウンロードしたファイルを実行して、SQLite をインストールします。
SQLite インストール後、Visual Studio のパッケージマネージャーコンソールを開きます。
パッケージマネージャーコンソールで以下のコマンドを実行すると、
SQLite にテーブルが作成されます。

update-database

図 7.1-1 パッケージマネージャーコンソール



7.2. 処理内容

サンプルの処理内容について説明します。

7.2.1. コンストラクタ

説明：フォームの初期設定を行います。

実装手順：

1. ControllerModel のインスタンスを作成します。
引数には EdgeControllerAPIKey.yml が配置されているパスを設定します。

実装例：

```
InitializeComponent();
var currentDirectory = Directory.GetCurrentDirectory();
controllerModel = new ControllerModel(currentDirectory);
```

7.2.2. 初期設定

説明：コントローラの初期設定を行います。

実装手順：

Step1：ポーリングで実行するサービス実装の登録
5.1.29 SetServiceMethod を呼び出します。

SetServiceMethod メソッドの引数には順に、
・メッセージを受け取った場合のサービス
・生成リクエストを受け取った場合のサービス
・削除リクエストを受け取った場合のサービス
を渡します。

サービスがない場合は、null を渡します。
サービスが存在する場合は、以下の型のオブジェクトを用意します。

```
new Dictionary<string, Action<string, List<Dictionary<string, object>>>>();
```

Dictionary の string には、サービス実装の内部 ID
Action<string, List<Dictionary<string, object>>>には、各サービスを設定します。
作成した Dictionary のオブジェクトを、SetServiceMethod の引数として渡します。

Step2：カレンダーイベントで実行するプロセス実装の登録
5.1.28 SetProcessMethod を使用します。

Step3：コントローラの初期設定
5.1.8 InitialSetting を使用します。

InitialSetting メソッドの引数には順に、
・コントローラのアドレス
・ポーリングの周期
・コントローラ内部のサービス実装定義 json ファイルのパス
・コントローラ内部のデータ実装定義 json ファイルのパス
を渡します。

json ファイルの出力には、5.3.4 SaveJsonFileBySpecifiedFileName を使用します。
InitialSetting を呼び出した後は、5.2 ControllerRestUtil クラスのメソッドで説明した
任意の通信メソッドを使用できます。

実装例：

```
var currentDirectory = Directory.GetCurrentDirectory();
var servicePath = Path.Combine(currentDirectory, "serviceimplement.json");
var dataPath = Path.Combine(currentDirectory, "dataimplement.json");
if (string.IsNullOrEmpty(this.txtAddress.Text))
{
    MessageBox.Show("アドレスが空です。");
    return;
}
Uri outUri;
if (!Uri.TryCreate(this.txtAddress.Text, UriKind.Absolute, out outUri))
{
    MessageBox.Show("アドレスが不正です。");
    return;
}
// サービス実装の登録
var serviceImplementsMethodListDict = new Dictionary<string, Action<string,
Dictionary<string, List<Dictionary<string, object>>>>>();
```

```
serviceImplementsMethodListDict.Add(LOCAL_SERVICE_ID, StoreEnvironmentData);
// リクエストによって実行するサービス実装の登録
Dictionary<string, Action<string, string, string, string>> requestMethodDict = new
Dictionary<string, Action<string, string, string, string>>();
requestMethodDict.Add(LOCAL_SERVICE_ID, GetRequest);
controllerModel.SetServiceMethod(serviceImplementsMethodListDict, null,
requestMethodDict);
// カレンダーイベントによって実行するプロセス実装の登録
Dictionary<string, Action> calendarProcessDict = new Dictionary<string, Action>();
calendarProcessDict.Add(Properties.Settings.Default.CALENDAR_EVENT_ID,
CalendarEvent);
controllerModel.SetProcessMethod(calendarProcessDict);
if (rbtnDB.Checked)
{
    controllerModel.SetDataMethod(SaveDataToDB, GetDataId, null);
    this.dataRecordList = GetDataRecordList();
    this.tradeDataRecordList = new List<TradeDataRecord>();
}
else
{
    this.dataRecordList = GetDataRecordListFromFile();
    controllerModel.SetDataMethod(SaveDataToFile, GetDataId, null);
    this.tradeDataRecordList = new List<TradeDataRecord>();
}
controllerModel.InitialSetting(this.txtAddress.Text, (int)this.nudPollingRate.Value,
servicePath, dataPath);
MessageBox.Show("初期設定を行いました。");
```

7.2.3. ポーリング開始

説明：ポーリング処理を開始します。

実装手順：

Step1：ポーリング処理

メソッドを呼び出します。

7.2.2 Initial Setting で登録したサービス実装が実行されます。

実装例：

```
controllerModel.PollingStart();
```

7.2.4. サービス実装確認

説明：サイトで登録したサービス実装の ID を確認します。

実装手順：

Step1：5.1.5 GetServiceImplementation を呼び出して、

サービス実装情報一覧を取得します。

実装例：

```
this.serviceList = this.controllerModel.GetServiceImplementations();
```

7.2.5. サービス実装のローカル ID 設定

説明：サービス実装のローカル ID を任意の値に設定します。

実装手順：

Step 1 ：ServiceStatusList を作成します。

Step 2 ：5.1.2 PutServiceImplementationsStatusList を呼び出して、

サービス実装のローカル ID 設定を行います。

※各 ServiceStatus で必須の設定項目は、id、local_id、status です。

実装例：

```
controllerModel.PutServiceImplementationStatusList(this.currentServiceStatusList);
```

7.2.6. データ実装 ID 確認

説明：サイトで登録したデータ実装の ID を確認します。
実装手順：
Step1 ：5.1.4 GetDataImplementations を呼び出して、
データ実装情報一覧を取得します。

実装例：

```
DataList = controller.GetDataImplementations();
```

7.2.7. データ実装のローカル ID 設定

説明：データ実装のローカル ID を任意の値に設定します。
実装手順：
Step1：5.1.23 PutDataImplementationStatusList を呼び出して、
データ実装のローカル ID 設定を行います。
引数には DataStatus オブジェクトのリストを渡します。
※各 DataStatus オブジェクトで必須の項目は、id、local_id、status です。
実装例：

```
controllerModel.PutDataImplementationStatusList(DataStatusList);
```

7.2.8. 契約を確認する

説明：現在の契約内容を確認します。
実装手順：
Step1：5.1.5 GetContract を呼び出して契約一覧を取得します。
実装例：

```
this.contract = controllerModel.GetContract();
```

7.2.9. カレンダーを確認する

説明：カレンダー実装の内容を確認します。
実装手順：
Step1：5.1.4 GetCalendar を呼び出してカレンダー情報一覧を取得します。
実装例：

```
this.calendarRoot = this.controllerModel.GetCalendar();
```

7.2.10. データを送る

説明：Push 型でデータを送信します。

実装手順：

Step1：送信するデータを List<Dictionary<string, object>>の形式にまとめる。

Step2：5.1.2 PostTradeData を呼び出してデータの送信を行う。

実装例：

```
// データ件数分、送信データを作成
foreach (var data in targetSensorDataList)
{
    var retDic = CIOF_SDK.Util.TypeUtil.CreateSendData();
    retDic.Add("Temperature", data.Tempareture);
    retDic.Add("Humidity", data.Humidity);
    retDic.Add("CO2", data.CO2);
    retDic.Add("Time stamp", data.MeasureDate);

    sendContentsList.Add(retDic);
}

// 取引データを送信
string contractDataId = controllerModel.PostTradeData(targetContract.id, sendContentsList);
```

7.2.11. データを受信する。

説明：送信されたデータを受け取ります。

実装手順：

Step1：Polling 内でデータを受け取るメソッドを、5.1.3 SetServiceMethod で登録します。データを受信するメソッドは第 1 引数として登録します。

Setp2：Polling 中に Step1 で登録したメソッドの第 2 引数にデータがわたってきます。

実装例：

```
// サービス実装の登録
var serviceImplementsMethodListDict = new Dictionary<string, Action<string,
Dictionary<string, List<Dictionary<string, object>>>>>();
serviceImplementsMethodListDict.Add(LOCAL_SERVICE_ID, StoreEnvironmentData);

controllerModel.SetServiceMethod(serviceImplementsMethodListDict,null,requestMeth
odDict);

public void StoreEnvironmentData(string contractId, Dictionary<string,
List<Dictionary<string, object>>> contentDict)
{
```

7.2.12. リクエストを送信する。

説明：生成または削除のリクエストを送ります。

実装手順：

Step1：取引事業者間で定めたルールに従ってリクエスト文を作成します。

Step2：5.1.16 PostRequestParameterByServiceId を呼び出してリクエストを送信します。

実装例：

```
※デザイナにて
this.txtCondition.Text = Count=3,From = 2021/1/1,To=2021/12/31;

controllerModel.PostRequestParameterByServiceId(this.localId,this.cbxType.Text,this.tx
tCondition.Text);
```

7.2.13. 生成リクエストを受信する

説明：送信された生成リクエストを受信します。

実装手順：

Step1：Polling 内で生成リクエストを受け取るメソッドを、5.1.31 SetServiceMethod を呼び出して登録します。リクエストを受信するメソッドは第 2 引数として登録します。

Step2：Polling 中に Step1 で登録したメソッドの引数にデータがわたってきます。

実装例：

```
var requestServiceMethodListDict = new Dictionary<string, Action<string, string, string>>();
requestServiceMethodListDict.Add(Properties.Settings.Default.LOCAL_SERVICE_ID, DataSendReqAction);

controllerModel.SetServiceMethod(null,requestServiceMethodListDict, null);
```

7.2.14. 削除リクエストを受信する。

説明：送信された削除リクエストを受信します。

実装手順：

Step1：Polling 内で削除リクエストを受け取るメソッドを、5.1.31 SetServiceMethod を呼び出して登録します。リクエストを受信するメソッドは第 3 引数として登録します。

Step2：Polling 中に Step1 で登録したメソッドの引数にデータがわたってきます。

実装例：

```
Dictionary<string, Action<string, string, string, string>> requestMethodDict = new Dictionary<string,Action<string,string,string,string>>();
requestMethodDict.Add(LOCAL_SERVICE_ID, GetRequest);
controllerModel.SetServiceMethod(serviceImplementsMethodListDict,null1,requestMethodDict);
```

7.2.15. カレンダーイベントを実行する

説明：登録されたカレンダー情報に基づいてカレンダーイベントを実行します。

実装手順：

Step1：カレンダーイベントによって実行されるメソッドを 5.1.32 SetProcessMethod を呼び出して登録します。

Step2：5.1.8 InitialSetting が呼び出された後、カレンダーの情報に従って、Step1 で登録したメソッドが実行されます。

実装例：

```
// カレンダーイベントによって実行するプロセス実装の登録
Dictionary<string, Action> calendarProcessDict = new Dictionary<string, Action>();
calendarProcessDict.Add(Properties.Settings.Default.CALENDAR_EVENT_ID,
CalendarEvent);
controllerModel.SetProcessMethod(calendarProcessDict);
```

7.2.16. コントローラを停止する

説明：コントローラを停止します。

実装手順：

Step1：5.1.33 Stop を呼び出して、コントローラを停止します。

実装例：

```
controllerModel.Stop();
```