

INDUSTRIAL  
EDGE  
SOLUTION  
WITH  
HARD REALTIME  
CAPABILITIES

**RT-edge**

Micronet.Co,

**マイクロネット**  
**Micronet**

INDUSTRIAL REALTIME EDGE COMPUTERS

# SDAT Container

RT-edge Stream Data Service Container  
ユーザーズマニュアル






株式会社マイクロネット

<http://www.mnc.co.jp>

TEL: +81(0)299-90-1733

FAX: +81(0)299-92-8557

本書で使用するマークについて

	ノート: 操作方法や手順等の補足情報や注釈を説明しています。
	情報: 製品を利用する上で有効な豆知識となる説明をしています。
	警告: 製品仕様上注意が必要な事象について説明しています。

Windows、Visual Studio は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。  
INtime は、米国 TenAsys Corporation の登録商標です。  
TenAsys®, INtime®, eVM® and iRMX® are registered trademarks in USA of the TenAsys Corporation.  
その他、本書に記載されている会社名、商品名は、各社の商標または登録商標です。  
本書の内容を無断で転載することは禁止されています。  
本書の内容に関しては、予告なしに変更することがあります。あらかじめご了承ください。

## 目次

用語解説	5
関連資料	7
1. 概要	8
1.1. RT-edge とサービスコンテナ	8
1.2. SDAT サービスコンテナ	10
2. 仕様	12
2.1. 動作環境	12
2.2. 使用可能タグ数	12
2.3. SDAT ファイルフォーマット	12
■ ファイル構成	13
■ ファイルヘッダ	13
■ レコード	14
■ レコード構成	14
■ レコードバイナリ	16
2.4. RT-edge コンテナ設定情報の範囲	16
2.5. 2.1.0.0 より前のバージョンをご使用のお客様	17
3. コンテナ導入フロー	18
4. インストール	19
4.1. ファイル	19
4.2. ファイル配置	19
4.3. 起動設定	20
4.4. 動作確認	21
4.5. 終了設定	22
5. 設定概要	23
5.1. ECI(RT-edge コンテナ設定情報)設定	23
RT-edge Object 設定	24
コンテナプロパティ設定	25
6. SDAT サービスコンテナ設定	26
6.1. RT-edge Object 設定	26
■ タグ・コレクション定義	26
■ 周期設定	27
6.2. コンテナプロパティ設定	28
■ サービス動作自動スタート指定 (.AutoRun)	28
7. 動作確認	29
7.1. 動作確認手順	29
7.2. サンプルによる動作確認	30
8. サービスインジケータタグ	31
8.1. 一覧	31
8.2. サービス起動状態ステータス(.Status)	31

8.3. サービス異常状態ステータス(.Error) .....	31
8.4. サービス実行状態ステータス(.Run) .....	32
8.5. サービス実行カウンタ(.Live) .....	32
<b>9. サービスプロパティタグ .....</b>	<b>33</b>
9.1. 一覧 .....	33
9.2. SDAT ファイルサイズ (.FragmentSize) .....	34
9.3. 残り SDAT ファイルサイズ (.FileFreeSize) .....	34
9.4. 最新レコード構成番号 (.FormatNumber) .....	35
9.5. SDAT ファイルデータ終端 (.AddPointer) .....	35
9.6. 記録容量 (.CapacitySize) .....	36
9.7. 残り記録容量 (.CapacityFreeSize) .....	36
9.8. スループット計測回数 (.MeasureCount) .....	37
9.9. スループット計測結果 (.TotalRecTime) .....	37
<b>10. サービスメッセージ .....</b>	<b>38</b>
<b>11. RT-edge タグデータの妥当性について .....</b>	<b>38</b>
<b>12. SDAT ファイルへのアクセス .....</b>	<b>39</b>
12.1. ファイルのオープン .....	44
■ ファイル形式 .....	44
■ ファイル名 .....	44
■ ファイルサイズ .....	44
■ 保存先フォルダ .....	44
■ ファイル保存数 .....	44
■ ファイルヘッダ .....	45
■ レコード情報 .....	46
■ ファイル切替 .....	49
■ ファイル削除 .....	49
■ ファイルオープン時の注意事項 .....	49
■ ロック中のファイルかどうかの見分け方 .....	50
■ ファイルのオープン .....	50
12.2. ファイルの読み込み .....	51
■ 読み込み位置の初期化 .....	51
■ ファイルヘッダの読み込み .....	51
■ レコードの読み込み .....	52
■ レコード情報(レコード構成/レコードバイナリ)判定 .....	52
■ ファイルの終端判定 .....	55
12.3. ファイルのクローズ .....	55
<b>13. トラブルシューティング .....</b>	<b>56</b>
13.1. サービスインジケータを確認すると.Run が false になっています。 .....	56
■ 原因: サービス開始要求を受け付けられていません。 .....	56
■ 対応: サービス開始要求を行います。 .....	56
13.2. RT-edge オブジェクトブラウザにサービスインジケータが表示されません。 .....	56
■ 原因: SDAT サービスコンテナ初期化処理中です。 .....	56
■ 対応: 終了させ、10 秒程度待ったのち再度 RT-edge オブジェクトブラウザを起動させます。 .....	56
13.3. SDAT ファイルにタグデータが保存されません。 .....	56
■ 原因: タグ設定がされていません。 .....	56

■ 対応:タグ設定を行います。.....	56
----------------------	----

## 用語解説

本ドキュメントにおいて使用される用語・略称について説明します:

表 1. 用語集

用語	説明
RT-edge	エッジコンピューティングを軸とする IT の情報処理と、FA における装置・機器の制御を融合し、密度の高い高頻度データ利用を可能とするソフトウェアプラットフォームです。FA で要求されるハードリアルタイム制御を組み込むことで、情報処理と機器・装置制御を可能とするエッジコントローラを構成することができます。
RT-edge 基本ソフトウェア	RT-edge 機能の核となる機能・ライブラリを実装するパッケージソフトウェア製品です。
IoT ゲートウェイ	IoT において、端末とインターネットを介した遠隔サーバー(クラウド)がデータのやりとりをする際、中継する役割を担う機能。サーバーや送信経路であるインターネット負荷の軽減をします。
IT システム	オンプレミスもしくはクラウドを活用した業務システムやアプリケーション。
INtime	INtime for Windows: Windows と協調動作可能なリアルタイムカーネル拡張ソフトウェアです(RTOS ソフトウェア)。 INtime Distributed RTOS(dRTOS): Windows OS を必要とせず、スタンドアロンで動作するリアルタイム OS です。
RTA	RealTime Application: リアルタイムアプリケーションの略称。INtime 上で動作するローダブルプロセスの拡張子です。INtime 上で動作するローダブルアプリケーションは、RTA という拡張子を持ちます。
RSL	Realtime Shared Library: リアルタイム共有ライブラリの略称。INtime 上でアプリケーションがロード可能なライブラリです。Windows 上で使用される DLL(Dynamic Link Library)のようなものです。RTA から使用されるライブラリインタフェース等は、こちらを使用して作成することができます。
API	Application Programming Interface: アプリケーションプログラミングインタフェースの略称。RT-edge ではデバイスへのアクセスインタフェースとして API ライブラリを提供しています。
NTX	INtime's Windows NT extension API: INtime 用 Windows NT 拡張 API の略称。NTX 関数は Windows プログラムが INtime リアルタイム環境上で実行するリアルタイムプログラムと通信を可能とする関数セットです。
OPC	主に産業オートメーション分野においてデータ交換を目的とした相互運用標準規格。
OPC UA	OPC UA(OPC Unified Architecture の略) 異なるプラットフォーム間のデータ交換を可能とした信頼性のある産業用通信データ交換標準。インダストリー4.0 の RAMI モデルに採用された規格。
エッジアプリケーション	RT-edge 内コンテナにより集積されたデータ(RT-edge Object)を活用、処理実行するソフトウェアです。
エッジコンピューティング	RT-edge 内で稼働する制御コンテナソフトウェアにより装置・機器から収集した高密度なデータをリアルタイムに収集、分析、フィードバックします。IT システムとの情報連携。
オンプレミス	サーバーやソフトウェア等の情報システム、アプリケーション等のソフトウェアを管理する施設内に設置して運用すること。
クラウド	サーバーやストレージ等のインフラやソフトウェアを必要とせず、必要な IT リソースが、インターネットを通じてオンデマンドで得られる形態、サービス。
産業用 PC	高信頼性、耐環境性、長期供給等の特徴をもつ産業用途の PC。
データ収集	診断、分析を行う対象となるデータを集積する処理。
データ加工	集積されたデータを利用しやすい形に変更する処理。
産業機器通信インターフェース	各種フィールドバス経由で機器、装置との通信、もしくは直接入出力デバイスの制御を行うインターフェースです。本インターフェースを介し、センサー値の参照やアクチュエータ制御が可能です。
サービス/EgService	RT-edge システムを構成する機能プロセス(rta/exe)です。
タグ/EgTag	瞬時値データ値 1 つを示すオブジェクトです。ユニーク名とグローバルなスコープを持ち、全ての EgService から読み書きが許されたオブジェクトです。タグは生成時にデータ

用語	説明
	型が確定され変更はできません。
リンクタグ	同一名称のタグを重複生成した場合に自動的に別名称で生成されるタグを指します。 通常のタグと同様、グローバルなスコープを持ち、全ての EgService から読み書きが許されたオブジェクトです。一つのタグに対し、異なるプロパティ情報を定義したい場合に使用します。
データセット/EgDataset	タグ 1 つ以上の組み合わせでデータ並び順(データ構造)を定義する名前付きオブジェクトです。
コレクタ/EgCollector	データセットに定義されたデータ構造に従って、同時刻のバイナリデータ列で生成し、データレコードとしてメールボックスに送信するオブジェクト（スレッド）です。
メールボックス/EgMailBox	時系列なデータセット、または時系列メッセージを FIFO で蓄えることができ、また受信イベントとして処理できるオブジェクトです。
タグ参照/TagRef	タグの参照として使用するオブジェクトです。タグの名前を保持し値は保持しません。サービスコンフィグファイルでデータセットの収集用タグとして定義することや、サービス内のオブジェクトとして定義することでサービスのメンバ変数として使用することができます。
コレクタ参照/CollectorRef	コレクタの参照として使用するオブジェクトです。コレクタの名前を保持しそれ以外のオブジェクトは保持しません。サービスコンフィグファイルでサービス内のオブジェクトとして定義することでサービス内のメンバ変数として使用することができます。
メッセージ	メールボックスで扱われる 1 レコード分のデータ、またはサービス間のコマンド、応答の電文です。
フレームワーク	フレームワークは、アプリケーションが API を組み合わせて実装するよくある処理についてマクロ化、自動化したものでサービスコンフィグファイルの記述により自動処理させることができます。
RT-edge コンテナ設定情報 (ECI)	RT-edge コンテナが RT-edge Object として展開する入出力データ定義の他、RT-edge コンテナフレームワークが、オブジェクト生成やコンテナサービス等自動処理するための定義設定情報(XML 型式)。
入力	RT-edge システムを中心に見た場合、外部の情報を RT-edge システムへ取り込む方向性のデータの流れを意味します。
出力	RT-edge システムを中心に見た場合、RT-edge システムが持つデータを外部に書き出す方向性のデータの流れを意味します。
RTCD	Realtime Common Data の略称。RT-edge システム上で最もベースとなる共有データ構造機能です。
RT-edge Object	RT-edge システム上で使用可能なオブジェクト群（機能群）の総称です。 例えば、センサーや装置から収集したデータをアプリケーション間で受け渡しを行う場合に使用するタグ、アプリケーション間でメッセージのやり取りを行う場合のメールボックス等、アプリケーション間でデータの受け渡しを行うケースにおいて利用されるオブジェクトです。 RT-edge Object は Windows アプリケーション間、INtime®アプリケーション間、Windows-INtime®アプリケーション間いずれの場合も利用可能です。
DB	DataBase(データベース)の略称。コンピュータ上で集積・整理された情報群。特定の条件に当てはまるデータを複数集め、使いやすい形に整理した情報のかたまり。
レコード	複数のタグ(RT-edge Object)を一括りにしたものです。
レコード構成	レコードに設定されたタグ定義情報(タグ名、タグタイプ、タグサイズ)を表します。
レコードバイナリ	レコードに設定されたタグデータを表します。
レコード構成番号	レコード構成とレコードバイナリをリンクさせる番号を表します。

## 関連資料

### RT-edge 製品に含まれる資料

表 2 .RT-edge 関連資料

名称	ファイル名	内容
RT-edge ユーザーズマニュアル	DOCRTEGEUSER.pdf	RT-edge システム全般的な内容の説明が記載されています。
RT-edge API リファレンス	DOCRTEGEAPI.pdf	RT-edge API の使用方法が記載されています。
RT-edge コンテナ作成マニュアル	DOCRTEGSRV.pdf	RT-edge コンテナの構造、サンプルプロジェクトを利用した作成方法等について記載されています。



# 1. 概要

## 1.1. RT-edge とサービスコンテナ

RT-edge とは、エッジコンピューティングを軸とする IT の情報処理と、FA における装置・機器の制御を融合し、密度の高い高頻度データ利用を可能とするソフトウェアプラットフォームです。

RT-edge の利用により、装置やセンサーからの高密度なデータ収集、分析だけでなく、提供される開発ライブラリキットを使用し、タグデータをレジスタとした機器制御を行うハードリアルタイムエッジアプリケーションの開発が可能です。

### サービスコンテナ

RT-edge の処理ターゲットは、エッジコンピューティングを軸とした IT 情報処理(IT-Process)と、ミリ秒精度のハードリアルタイム性を要求される FA 制御(FA-Control)に分類され、ターゲットの機能に特化した専門処理サービスをコンテナ(サービスコンテナ)と呼びます。

IT 情報処理ターゲットは上位層にあり、主に外部システムからの要求指示の受付や、外部システムへのデータ公開、通信等を担う要素となります。IT 情報処理サービスコンテナは、制御システムのコンソール画面や外部システムから WEB ブラウザ経由でのアクセス機能、制御データ情報を外部クラウドストレージに保存する機能等、上位システムとの接続・インターフェースを提供します。

一方、FA 制御ターゲットは下位層に位置し、主に通信やハードウェアへの直接 I/O 入出力等により装置・機器制御を担う要素です。FA 制御サービスコンテナは、産業用フィールドバスやコントローラ通信プロトコルによるロボット制御、計測機器からのデータロギング、デジタルパルス出力等、装置・機器へのアクセスを提供します。

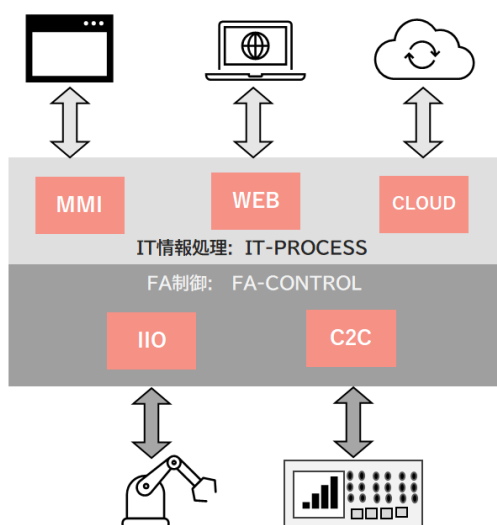


図 1. ターゲットとコンテナ

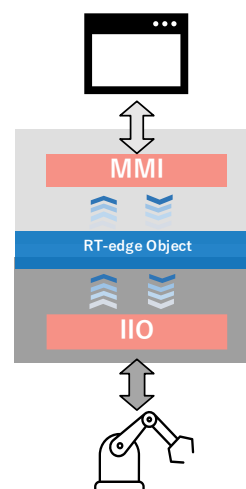


図 2. コンテナの役割



### サービスコンテナ例

WEB: IT 情報処理ターゲット内には、IIS(Internet Information Service)を紹介し、制御情報をインターネット上に公開する WEB サービスコンテナ  
 IIO: ロボットアーム制御に特化した産業 I/O サービスコンテナ

サービスコンテナはターゲットに特化した入出力データを RT-edge Object であるシステム内でグローバルにアクセス可能なタグ情報としてリンクし、このタグ情報のコレクションを公開します。

サービスコンテナは、タグ情報コレクションや、動作・挙動を決定するパラメータ設定と、ターゲット処理に特化した一つ以上の実行処理の集合体です:

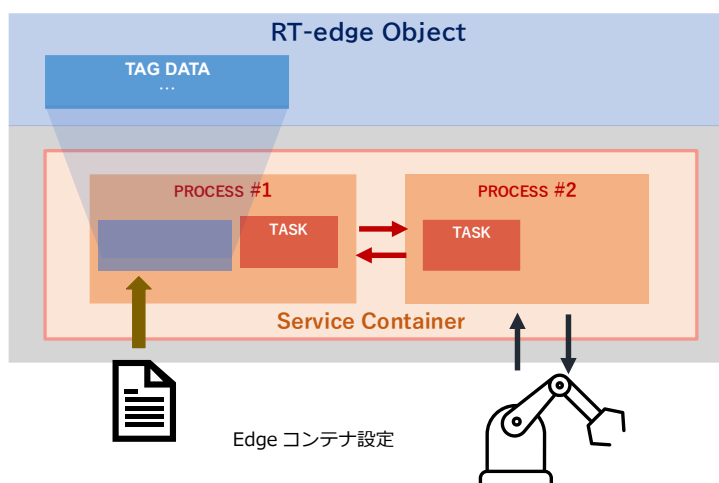


図 3. サービスコンテナの構造

### サービスコンテナ例

#### PROCESS #1

RT-edge Object を使用する処理プロセス (サービスハンドラ)

#### PROCESS #2

ターゲット特化処理プロセス

#### Edge コンテナ設定情報

タグデータコレクション等コンテナ設定

## 1.2. SDAT サービスコンテナ

SDAT サービスコンテナは、コンシューマ型のコンテナとして動作し、タグ(RT-edge Object)をレコードにして、特定のフォーマットに沿って生成されたバイナリファイル(SDAT ファイル)に保存します。

SDAT ファイルには同時刻のレコードが保存されるため、SDAT コンテナを導入することで、システムの状況解析を行うことができます。

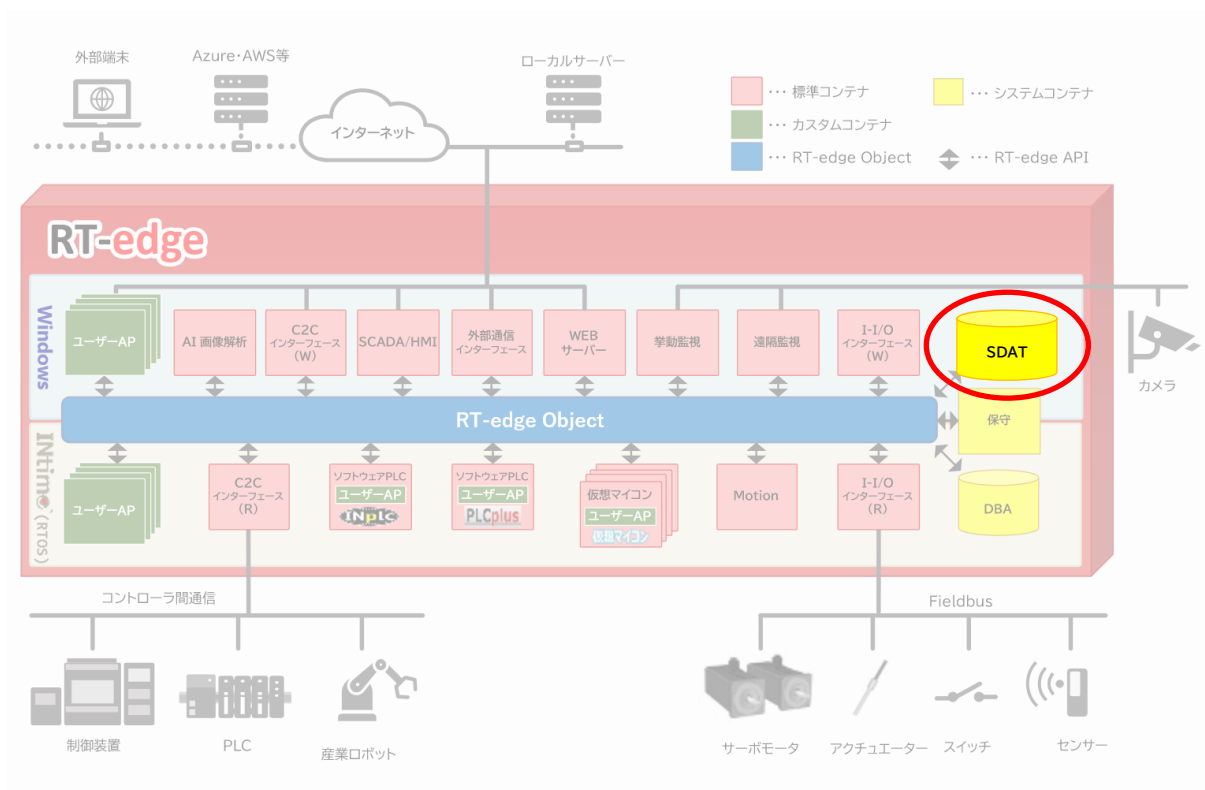


図 4.RT-edge 相関図

構成要素

SDAT コンテナは以下コンポーネントから構成されます:

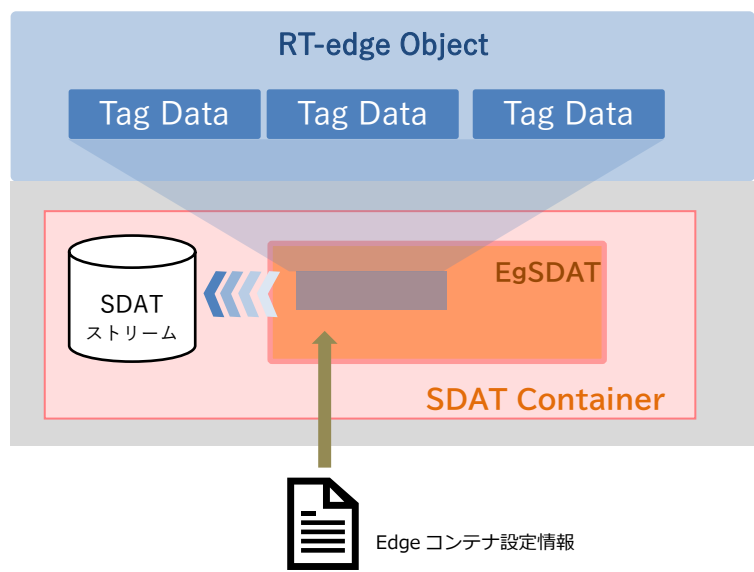


図 5. SDAT コンテナ構成

表 3.SDAT コンテナ構成要素

コンポーネント	内容
EgSDAT	タグデータを SDAT ファイルに保存する処理が含まれます。 EgSDAT は RT-edge コンテナ設定情報 (ECI) で設定した複数のタグを 1 レコードとして構成し、レコード単位で SDAT ファイルへ保存します。 またコンテナ自身の状態の監視を行います。
SDAT ストリーム	ストリームバイナリ形式のファイルで、タグデータなどが含まれます。
RT-edge コンテナ設定情報 (ECI)	SDAT ファイルに保存するタグや周期を設定します。



本ドキュメントでは、主に SDAT コンテナの利用方法について説明します。RT-edge 基本ソフトウェア、他サービスコンテナについては各々のマニュアルを参照ください。

## 2. 仕様

### 2.1. 動作環境

- RT-edge 基本ソフトウェアバージョン 3.5.3 以降



その他、実行環境についての制限、RT-edge 基本ソフトウェアに準拠します。

### 2.2. 使用可能タグ数

レコード対象となるタグをデータセットに定義します。

他のコンテナが使用する入力タグ参照(TagRefs\_IN),出力タグ参照(TagRefs\_OUT)は SDAT コンテナでは使用していません。

またデータセットに定義可能なタグ数は、RT-edge 基本ソフトウェア仕様に依存します。

### 2.3. SDAT ファイルフォーマット

SDAT ファイルは、以下の表 4. SDAT ファイル仕様で保存されます。

表 4. SDAT ファイル仕様

項目	内容
ファイル形式	バイナリ形式
ファイル名	yyyyMMddHHmmss_SDAT.SDT (yyyyMMddHHmmss：ファイル生成日時)
ファイルサイズ	512MB <ul style="list-style-type: none"> <li>● ファイル生成時にファイルサイズを確保します。</li> <li>● データ書き込み時、書き込むデータサイズが残りファイルサイズより上の場合は、SDAT ファイルを切り替えます。</li> </ul>
収納パス	SDAT コンテナのカレントパス
収納サイズ(記録容量)	4GB <ul style="list-style-type: none"> <li>● SDAT ファイル切り替え時、残り収納サイズがない場合はファイル名が古い日時のファイルを削除します。</li> </ul>



SDAT コンテナ起動中は SDAT ファイルをロックしているため、SDAT コンテナ起動中に SDAT ファイルを参照することができません。  
SDAT コンテナ停止後、SDAT ファイルを参照ください。

## ■ ファイル構成

SDAT ファイルは、ファイルヘッダ、レコードで構成されます。

0x00000000	<u>ファイルヘッダ(32 バイト)</u>
:	ファイル識別子、最新のレコード構成番号、データ終端などが含まれます。
0x0000001F	
0x00000020	<u>登録済みレコード：レコード構成/レコードバイナリ</u>
:	タグ情報を示す「レコード構成」、タグデータを示す「レコードバイナリ」で分類されます。
:	
:	<u>レコード書き込み領域</u>
0x1FFFFFFF	レコード登録時に使用します。

図 6.ファイル構成

## ■ ファイルヘッダ

ファイルヘッダは 32 バイトで構成されます。

0x00000000	<u>SDAT ファイル識別子(4 バイト)</u>
:	SDAT ファイルを示す識別子。
0x00000003	Shift_JIS コードで“SDAT”が設定されます。
0x00000004	<u>ファイルヘッダサイズ(1 バイト)</u>
	ファイルヘッダサイズ(32)がバイト単位で設定されます。
0x00000005	<u>最新のレコード構成番号(4 バイト)</u>
:	レコード構成登録時のレコード構成番号。
0x00000008	レコード構成登録後インクリメントされます。
0x00000009	<u>データの終端(4 バイト)</u>
:	レコードの収納最終位置。
0x0000000C	レコード登録時にインクリメントされます。
0x0000000D	(予備) 将来用に確保しております。
:	
0x0000001F	

図 7.ファイルヘッダ構成

## ■ レコード

レコードは、同時刻の複数のタグデータをまとめた構造です。

レコードには、タグ情報を収納した「レコード構成」、タグデータを収納した「レコードバイナリ」の 2 種類で分類され、各レコードの見分け方は保存された前のレコードで判定します。

表 5. 登録レコードの判別

ケース	登録レコード
ファイルヘッダの次のレコード	レコード構成
保存されていないレコード構成番号が格納されたレコード	レコード構成
コンテナ起動時、初回登録されたレコード	レコード構成
上記以外	レコードバイナリ

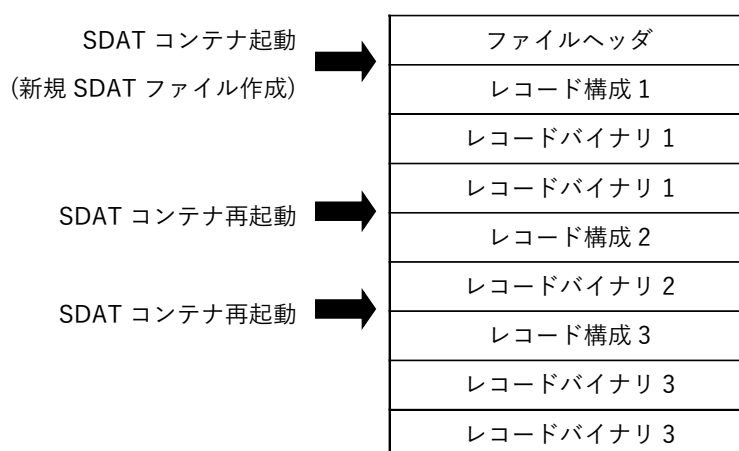


図 8. SDAT ファイルイメージ

## ■ レコード構成

レコード構成は、レコード構成サイズ、レコード構成番号、タグ情報で構成されます。

0x00000000	<u>レコード構成サイズ(2 バイト)</u>
0x00000001	レコード構成のバイトサイズが設定されます。
0x00000002	<u>レコード構成番号(4 バイト)</u>
:	登録時のレコード構成番号が設定されます。
0x00000005	
0x00000006	<u>タグ数(2 バイト)</u>
0x00000007	レコード構成に登録されたタグ数(1~n)が設定されます。
0x00000008	<u>タグ 1. タグ名長さ(1 バイト)</u>
	レコード構成に登録されたタグ 1 の長さ (NULL は除く) がバイト単位で設定されます。
0x00000009	<u>タグ 1. タグ名(タグ 1 タグ名長さで構成)</u>
:	レコード構成に登録されたタグ 1 のタグ名 (NULL は除く Shift_JIS コード) が設定されます。

:	<u>タグ 1.タグタイプ(1 バイト)</u>
:	レコード構成に登録されたタグ 1 のタグタイプが設定されます。
:	<u>タグ 1.タグサイズ(1 バイト)</u>
:	レコード構成に登録されたタグ 1 のタグサイズが設定されます。
:	<u>タグ 2 からタグ n-1 までのタグ情報</u>
:	タグ 2 からタグ n-1 までのタグ情報（タグ名長さ、タグ名、タグタイプ、タグサイズ）が設定されます。
:	<u>タグ n.タグ名長さ(1 バイト)</u>
:	レコード構成に登録されたタグ n の長さ（NULL は除く）がバイト単位で設定されます。
:	<u>タグ n.タグ名(タグ n タグ名長さで構成)</u>
:	レコード構成に登録されたタグ n のタグ名（NULL は除く Shift_JIS コード）が設定されます。
:	<u>タグ n.タグタイプ(1 バイト)</u>
:	レコード構成に登録されたタグ n のタグタイプが設定されます。
:	<u>タグ n.タグサイズ(1 バイト)</u>
:	レコード構成に登録されたタグ n のタグサイズが設定されます。

図 9.レコード構成

表 6. タグタイプ(型)、タグサイズ（RT-edge ユーザーズマニュアルより抜粋）

No	型	サイズ(byte)	用途
0	UNDEFINE	-	未定義
1	Boolean	1	bool 値
2	SByte	1	符号付き 8 ビット整数
3	Byte	1	符号なし 8 ビット整数
4	Int16	2	符号付き 16bit 整数
5	UInt16	2	符号なし 16bit 整数
6	Int32	4	符号付き 32bit 整数
7	UInt32	4	符号なし 32bit 整数
8	Int64	8	符号付き 64bit 整数
9	UInt64	8	符号なし 64bit 整数
10	Float	4	単精度実数(32bit)
11	Double	8	倍精度実数(64bit)

例えば、タグ 1 に「Tag Name="Tag1" Type=6 Size=4」のタグを設定した場合、レコード構成のタグ 1 には、以下のように配置されます。

オフセット 0	0x04	"Tag1"の文字の長さ
+1	'T'	タグ名
+2	'a'	
+3	'g'	
+4	'1'	



+5	0x06	タグタイプ
+6	0x00	
+7	0x04	タグサイズ
+8	0x00	

図 10.タグ情報配置例

## ■ レコードバイナリ

レコードバイナリには、レコードバイナリサイズ、レコード構成番号、タグデータで構成されます。

0x00000000	<u>レコードバイナリサイズ(2 バイト)</u>
0x00000001	レコードバイナリのバイトサイズ(バイト単位)が設定されます。
0x00000002	<u>レコード構成番号(4 バイト)</u>
:	レコードバイナリが属するレコード構成のレコード構成番号が設定されます。
0x00000005	
0x00000006	<u>タグ 1.データ(タグ 1 に定義されたタグサイズで構成)</u>
:	レコード構成に定義されたタグ 1 のタグデータが設定されます。
:	<u>タグ 2 からタグ n-1 までのタグデータ</u>
:	レコード構成に定義されたタグ 2 からタグ n-1 までのタグデータが設定されます。
:	<u>タグ n.データ(タグ n に定義されたタグサイズで構成)</u>
:	レコード構成に定義されたタグ n のタグデータが設定されます。

図 11.レコードバイナリ構成

例えば、タグ 1「Tag Name="Tag1" Type=6 Size=4」のデータが 0x12345678 の場合、レコードバイナリのタグ 1 には、以下のように配置されます。

オフセット 0	0x78	タグ 1 データ
+1	0x56	
+2	0x34	
+3	0x12	

図 12.タグデータ配置例

## 2.4. RT-edge コンテナ設定情報の範囲

RT-edge コンテナ設定情報(ETI RT-edge Container Information: EGSDAT.XML)内の各プロパティには設定範囲が定められており、範囲外の値を指定すると初期化エラーとなります。詳細は「9. サービスプロパティタグ」をご参照ください。

## 2.5. 2.1.0.0 より前のバージョンをご使用のお客様

SDAT コンテナ バージョン 2.1.0.0 では、コンポーネント名を「EgSDB」から「EgSDAT」へ変更しました。そのため、2.1.0.0 より前のバージョンで作成された ECI ファイルを 2.1.0.0 以降で使用する場合、以下の内容を更新してください。

表 7 .更新内容

項目	内容
起動設定	1) EgBoot.xml に定義したコンテナ名を「EgSDB」から「EgSDAT」に変更します。
終了設定	1) EgShDown.xml に定義したコンテナ名を「EgSDB」から「EgSDAT」に変更します。
ECI ファイル名	1) ECI ファイルを「EgSDB.xml」から「EgSDAT.xml」に変更します。
データセット名	1) ECI ファイルに定義したデータセット名を「SDBset」から「SDATset」に変更します。
コレクタ名	1) ECI ファイルに定義したコレクタ名を「SDBcol」から「SDATcol」に変更します。
コンテナプロパティ	1) ECI ファイルに定義したサービス自動スタート指定(.AutoRun)のタグ名を「SERVICE.EgSDB.AutoRun」から「SERVICE.EgSDAT.AutoRun」に変更します。

## 3. コンテナ導入フロー

以下の手順にて SDAT サービスコンテナの導入を行ってください:

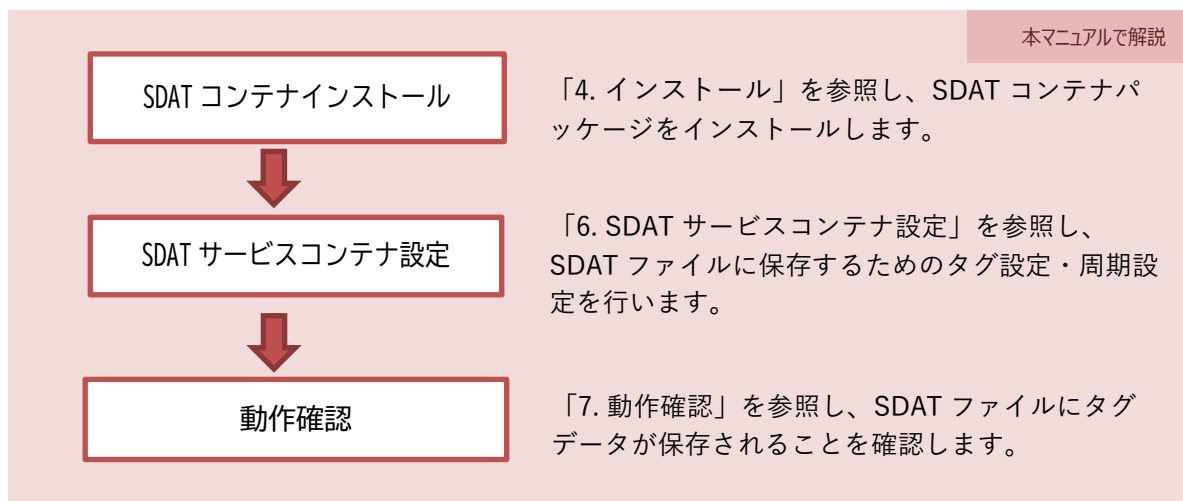


図 13.SDAT サービスコンテナの導入

## 4. インストール

### 4.1. ファイル

SDAT サービスコンテナパッケージには以下のファイルコンポーネントが含まれています:

表 8.SDAT サービスコンテナパッケージコンポーネント

フォルダ階層	ファイル名	説明
RT-edge\	EgSDAT.exe EgSDAT.exe.config	SDAT サービスコンテナにおいて、タグ(RT-edge Object)をSDAT ファイルに保存するサービスプロセスです。
RT-edge\	EgSDAT.xml	SDAT サービスコンテナ用 RT-edge 設定情報です。本設定ファイルの構成により SDAT ファイルに保存するタグの設定、保存周期の設定を行います。

### 4.2. ファイル配置

SDAT サービスコンテナパッケージは、ファイルコンポーネントを ZIP 圧縮した形式で配布されます。配布された SDAT サービスコンテナパッケージ(RTEdgeSDAT.zip)を、RT-edge ディレクトリ内へ解凍します。

以下の図 14.SDAT サービスコンテナコンポーネント配置イメージは、C ドライブ直下の RT-edge フォルダに展開した場合のイメージ図です。

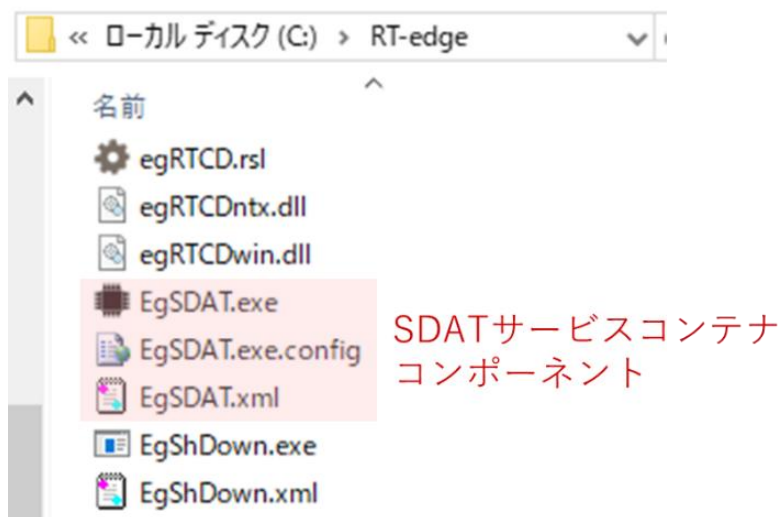


図 14.SDAT サービスコンテナコンポーネント配置イメージ

### 4.3. 起動設定

RT-edge におけるサービスコンテナ、および関連サービス・アプリケーションの設定は、RT-edge ブートストラッパー設定により行います。SDAT サービスコンテナの起動設定も同様、RT-edge ブートストラッパー設定に準拠します:

- 1) (RT-edge インストールパス)\EgBoot.xml をテキストエディタで開きます。
- 2) RTedge エlement内の Services Element内に、SDAT サービスコンテナ用のElement(Service Element)を追加します。

```
<Service Name="EgSDAT" Path="EgSDAT.exe" >
</Service>
```

Service Name="EgSDAT" SDAT サービスコンテナ登録名

Path="EgSDAT.exe" SDAT サービスコンテナのファイル名(EgSDAT.exe)を指定します。

- 3) 編集を保存し、ファイルを閉じます。
- 4) 追加結果は以下のようになります。

```
<?xml version="1.0" encoding="utf-8"?>
<RTedge xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
:
<Services>
<Service Name ="EgBoot" Argument="RTCD=NodeA;TagMaxNum=10000" >
</Service>
<Service Name="EgLog" Path="EgLog.exe" Argument="DispNumMax=500" >
</Service>
<Service Name="EgTime" Path="EgTime.exe" >
</Service>
<Service Name="EgSDAT" Path="EgSDAT.exe" >
</Service>
</Services>
</RTedge>
```

図 15.SDAT コンテナ起動登録



SDAT コンテナは、他のサービスコンテナのタグデータを参照するため、最後に起動するよう追加してください。

## 4.4. 動作確認

RT-edge ソフトウェアを起動し、SDAT サービスコンテナが正常に起動することを確認します:

- 1) RT-edge ソフトウェアを開始します(開始方法は RT-edge ユーザーズマニュアルをご参照ください)。
- 2) RT-edge オブジェクトブラウザ((RT-edge インストールパス) \EgBrow.exe)を起動します。



RT-edge 起動直後に RT-edge オブジェクトブラウザを起動すると初期化中の為、想定されるタグが表示されない場合があります。一度 RT-edge オブジェクトブラウザを終了し、再度起動させてください。

- 3) サービスインジケータタグから、正常状態であることを確認します(サービスインジケータタグの参照方法は、8. サービスインジケータタグを参照ください)。

### 正常状態

- SERVICE.EgSDAT.Status が 01(1) であること
- SERVICE.EgSDAT.Run が true(1) であること
- SERVICE.EgSDAT.Error が False(0) であること
- SERVICE.EgSDAT.Live が 増加していくこと

RT-edge Object Browser

ファイル

Tags	Containers	Collectors	Datasets	Mailboxes
Name	Current Value	Type	Source	
SERVICE.EgSDAT	00000000 (0)	Int32		
SERVICE.EgSDAT.Status	01 (1)	byte		
SERVICE.EgSDAT.Error	False (0)	bool		
SERVICE.EgSDAT.Run	True (1)	bool		
SERVICE.EgSDAT.Live	00000030 (48)	UInt32		

図 16. RT-edge オブジェクトブラウザ起動時の様子

上記の状態になっていない場合には、以下のトラブルシューティングをご参照ください。

「13.1. サービスインジケータを確認すると.Run が false になっています。」

## 4.5. 終了設定

RT-edge におけるサービスコンテナ、および関連サービス・アプリケーションの終了は、RT-edge 終了サービス「EgShDown」により行います。SDAT サービスコンテナの終了設定も同様、RT-edge 終了サービス設定に準拠します:

- 1) (RT-edge インストールパス)¥EgShDown.xml をテキストエディタで開きます。
- 2) ArrayOfAnyType エレメント内に、SDAT コンテナを追加します。

```
<anyType xsi:type="xsd:string">EgSDAT</anyType>
```



SDAT コンテナを追加する際は、他のサービスコンテナのタグデータを参照するため、最初に終了するよう追加してください。

- 3) 編集を保存し、ファイルを閉じます。
- 4) 追加結果は以下のようになります。

```
...:
<ArrayOfAnyType ...>
  <anyType xsi:type="xsd:string">EgSDAT</anyType>
  <anyType xsi:type="xsd:string">EgLog</anyType>
  <anyType xsi:type="xsd:string">EgTime</anyType>
  <anyType xsi:type="xsd:string">EgBoot</anyType>
</ArrayOfAnyType>
```

図 17.SDAT コンテナ終了登録

RT-edge ソフトウェアの終了は EgShDown.exe を実行します。  
EgShDown については RT-edge ユーザーズマニュアルを参照ください。

## 5. 設定概要

サービスコンテナ設定により担当するターゲットのデータとタグとの接続が可能となります。  
サービスコンテナにおける基本設定は、タグ、データセットの定義を主とした RT-edge Object 設定と、サービスコンテナの入出力周期やプライオリティ設定等、コンテナプロパティ設定に分類されます：

表 9. サービスコンテナにおける基本設定

設定項目	説明
<b>RT-edge Object 設定</b>	<b>タグ設定</b> ローカルタグ生成設定・リンクタグ生成設定 タグ参照設定 <b>データセット設定</b> タグ・コレクション定義 周期・プライオリティ設定
<b>コンテナプロパティ設定</b>	データ更新方式(オンデマンド・サイクリック(周期設定)) プライオリティ設定等 ※コンテナプロパティ値は各サービスコンテナにより実装が異なります。

### 5.1. ECI(RT-edge コンテナ設定情報)設定

RT-edge Object 設定、プロパティ設定は、コンテナ毎に定義する設定情報(ECI: RT-edge コンテナ設定情報)に基づきます。ECI ファイルは XML 形式のテキストファイルとして生成されています：

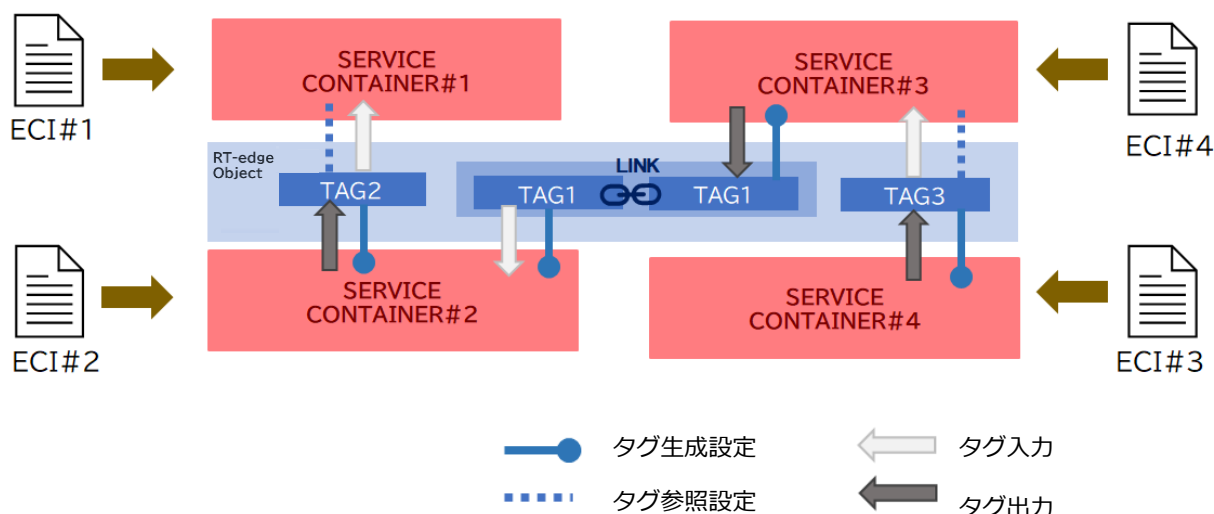


図 18. サービスコンテナと ECI

各サービスコンテナは、タグ・リンクタグ生成設定を行います。  
タグに対する入出力方向設定・参照設定を行います。



## RT-edge Object 設定

RT-edge Object 設定では、ECI ファイル内で編集する XML タグの編集要素は以下のように定義されています:

表 10. RT-edge Object 設定

設定項目	設定手順												
タグ設定	<p><u>ローカルタグ</u></p> <p>&lt;Tags&gt;エレメント内に、&lt;Tag&gt;を生成します。</p> <p>サービスコンテナ独自の名称(一意名)で&lt;Tag Name=&gt;の設定を行います。</p> <p><u>リンクタグ</u></p> <p>&lt;Tags&gt;エレメント内に、&lt;Tag&gt;を生成します。</p> <p>他サービスコンテナの提供するタグと同名で&lt;Tag Name=&gt;の設定を行います。</p> <p><b>Tag</b></p> <table> <tr> <th>キーワード</th><th>説明</th></tr> <tr> <td>Name</td><td>公開するタグ名を設定します。</td></tr> <tr> <td>Type</td><td>RT-edge データ型に関連する型定義値を設定します。 ※参照: データ型</td></tr> <tr> <td>Size</td><td>タグデータサイズを指定します。 ※参照: データ型</td></tr> <tr> <td>Address</td><td>サービスコンテナにおけるデータ取得元、宛先となるアドレス情報を指定します。 本アドレス書式は、サービスコンテナ毎に異なります。</td></tr> <tr> <td>Comment</td><td>タグに対するコメントを設定します。</td></tr> </table>	キーワード	説明	Name	公開するタグ名を設定します。	Type	RT-edge データ型に関連する型定義値を設定します。 ※参照: データ型	Size	タグデータサイズを指定します。 ※参照: データ型	Address	サービスコンテナにおけるデータ取得元、宛先となるアドレス情報を指定します。 本アドレス書式は、サービスコンテナ毎に異なります。	Comment	タグに対するコメントを設定します。
キーワード	説明												
Name	公開するタグ名を設定します。												
Type	RT-edge データ型に関連する型定義値を設定します。 ※参照: データ型												
Size	タグデータサイズを指定します。 ※参照: データ型												
Address	サービスコンテナにおけるデータ取得元、宛先となるアドレス情報を指定します。 本アドレス書式は、サービスコンテナ毎に異なります。												
Comment	タグに対するコメントを設定します。												
参照	<p><u>入力参照</u>: &lt;TagRefs_IN&gt; エレメント内</p> <p><u>出力参照</u>: &lt;TagRefs_OUT&gt; エレメント内</p> <p>&lt;TagRef Name=&gt;に参照するタグを指定します。</p> <p><b>TagRef</b></p> <table> <tr> <th>キーワード</th><th>説明</th></tr> <tr> <td>Name</td><td>参照するタグを指定します。</td></tr> </table>	キーワード	説明	Name	参照するタグを指定します。								
キーワード	説明												
Name	参照するタグを指定します。												
データセット設定	<p>&lt;Datasets&gt;エレメント内に、&lt;Dataset&gt;</p> <p>タグ・コレクション定義</p> <p>&lt;Dataset Name=&gt;にデータセット名を指定します。</p> <p>Dataset エレメント内に、&lt;TagRefs&gt;エレメントを作成します。</p> <p>&lt;TagRef Name=&gt;に参照するタグを指定します。</p> <p><b>TagRef</b></p> <table> <tr> <th>キーワード</th><th>説明</th></tr> <tr> <td>Name</td><td>参照するタグを指定します。</td></tr> </table> <p>※Dataset 内に TagRef オブジェクトを列挙します。</p>	キーワード	説明	Name	参照するタグを指定します。								
キーワード	説明												
Name	参照するタグを指定します。												

設定項目	設定手順										
周期・プライオリティ設定	<p>&lt;Collectors&gt;エレメント内に、&lt;Collector&gt;を作成し</p> <p>&lt;Collector Name=&gt;に名称を設定します (Dataset を収集する機能名)</p> <p><b>Collector</b></p> <table> <tr> <th>キーワード</th><th>説明</th></tr> <tr> <td>Name</td><td>Dataset 収集機能名を指定します</td></tr> <tr> <td>Interval</td><td>収集周期を指定します (1ms 単位)</td></tr> <tr> <td>Priority</td><td>プライオリティを設定します</td></tr> <tr> <td>DatasetName</td><td>収集するデータセット名を指定します。</td></tr> </table>	キーワード	説明	Name	Dataset 収集機能名を指定します	Interval	収集周期を指定します (1ms 単位)	Priority	プライオリティを設定します	DatasetName	収集するデータセット名を指定します。
キーワード	説明										
Name	Dataset 収集機能名を指定します										
Interval	収集周期を指定します (1ms 単位)										
Priority	プライオリティを設定します										
DatasetName	収集するデータセット名を指定します。										

## コンテナプロパティ設定

サービスコンテナプロパティ設定値は、Tag として登録されており、サービスコンテナ実装毎に数や種類は異なります。規定値プロパティは、SERVICE.キーワードをプリフィックスとしたタグ名で登録されています:

表 11.コンテナプライオリティ設定

設定項目	設定手順												
コンテナプロパティ値	<p>&lt;Tags&gt;エレメント内に、&lt;Tag&gt;において、SERVICE. キーワードをプリフィックスとしたタグは、コンテナプロパティタグです:</p> <p><b>Tag</b></p> <table> <tr> <th>キーワード</th><th>説明</th></tr> <tr> <td>Name</td><td>SERVICE. キーワードをプリフィックスとした名称で設定されています。</td></tr> <tr> <td>Type</td><td>RT-edge データ型に関連する型定義値を設定します。</td></tr> <tr> <td>Size</td><td>タグデータサイズを指定します。</td></tr> <tr> <td>Value</td><td>設定値</td></tr> <tr> <td>Comment</td><td>タグに対するコメントを設定します。</td></tr> </table>	キーワード	説明	Name	SERVICE. キーワードをプリフィックスとした名称で設定されています。	Type	RT-edge データ型に関連する型定義値を設定します。	Size	タグデータサイズを指定します。	Value	設定値	Comment	タグに対するコメントを設定します。
キーワード	説明												
Name	SERVICE. キーワードをプリフィックスとした名称で設定されています。												
Type	RT-edge データ型に関連する型定義値を設定します。												
Size	タグデータサイズを指定します。												
Value	設定値												
Comment	タグに対するコメントを設定します。												



コンテナプロパティ設定については、各サービスコンテナに付与するユーザズマニュアルを参照してください。

## 6. SDAT サービスコンテナ設定

SDAT コンテナにおける RT-edge Object 設定には、SDAT ファイルに保存するための設定（タグ、周期）が含まれます。

### 6.1. RT-edge Object 設定

SDAT サービスコンテナでは SDAT ファイルに保存するためのタグ設定、周期設定を行います。

#### ■ タグ・コレクション定義

SDAT コンテナはデータセットに定義されたタグをレコードとして、保存します。

SDAT ファイルに保存するタグの順番は、データセットに設定されたタグ順番と同じです。

使用するデータセットは ECI にデフォルトで定義されております。

```
...:
<Dataset Name = "SDATset">
  <TagRefs>
    <TagRef Name="EDGESYSTEM.CurrentTime" />
    <TagRef Name="EDGESYSTEM.CurrentTime_ms" />
    <TagRef Name="EDGESYSTEM.CurrentTime_us" />
  </TagRefs>
</Dataset>
```

図 19. データセットのデフォルト定義

表 12. データセット設定

データ項目	設定値	編集可/不可	説明
Dataset Name	SDATset	不可	SDAT コンテナが使用するコレクタ
TagRef Name	EDGESYSTEM.CurrentTime	可	現在時刻 1970.01.01 からの経過秒
TagRef Name	EDGESYSTEM.CurrentTime_ms	可	現在時刻ミリ秒
TagRef Name	EDGESYSTEM.CurrentTime_us	可	現在時刻マイクロ秒



現在時刻のタグは EgTime サービスが生成するため、EgTime サービスが起動されていることが前提です。  
EgTime サービスについては RT-edge ユーザーズマニュアルを参照ください。



データセットに設定可能なタグ数は RT-edge 基本ソフトウェアに準拠します。

#### タグの追加

例えば、他のコンテナ/サービスに定義されたタグ「TagTest\_0001」「TagTest\_0002」を新規に SDAT ファイルへ保存する場合は、SDAT ファイルに保存する TagRef エlementをデータセットに追加します。

```

...:
<Dataset Name = "SDATset">
  <TagRefs>
    <TagRef Name="EDGESYSTEM.CurrentTime" />
    <TagRef Name="EDGESYSTEM.CurrentTime_ms" />
    <TagRef Name="EDGESYSTEM.CurrentTime_us" />
    <TagRef Name="TagTest_0001" />
    <TagRef Name="TagTest_0002" />
  </TagRefs>
</Dataset>

```

図 20. タグ追加例

## タグの削除

例えば、現在時刻マイクロ秒のタグ「EDGESYSTEM.CurrentTime\_us」が不要な場合は、TagRef エlement をデータセットから削除します。

```

...:
<Dataset Name = "SDATset">
  <TagRefs>
    <TagRef Name="EDGESYSTEM.CurrentTime" />
    <TagRef Name="EDGESYSTEM.CurrentTime_ms" />
  </TagRefs>
</Dataset>

```

図 21. タグ削除例

## ■ 周期設定

SDAT コンテナはコレクタに定義された周期で保存します。

使用するコレクタは ECI にデフォルトで定義されております。

```

...:
<Service Name = "EgSDAT">
  <Collectors>
    <Collector Name="SDATcol" Interval="1000" Priority="140" DatasetName="SDATset"/>
  </Collectors>
  <CollectorRefs>
    <CollectorRef Name="SDATcol" />
  </CollectorRefs>
</Service>

```

図 22. コレクタのデフォルト定義

表 13. コレクタ設定

データ項目	設定値	編集可/不可	説明
Name	SDATcol	不可	SDAT コンテナが使用するコレクタ名
Interval	1000	可	SDAT ファイルへの保存周期(ミリ秒)
Priority	140	不可	SDAT コンテナが使用するスレッドプライオリティ
DatasetName	SDATset	不可	SDAT ファイルに保存する際のデータセット名



コレクタで編集可能な項目は「Interval」のみです。  
他の項目は編集しないでください。

### 周期の変更

例えば、2000 ミリ秒周期で SDAT ファイルに保存した場合は、Interval を"2000"に変更します。

```
...:
<Service Name = "EgSDAT">
  <Collectors>
    <Collector Name="SDATcol" Interval="2000" Priority="140" DatasetName="SDATset"/>
  </Collectors>
  <CollectorRefs>
    <CollectorRef Name="SDATcol" />
  </CollectorRefs>
</Service>
```

図 23. 保存周期の変更

## 6.2. コンテナプロパティ設定

SDAT コンテナは以下のプロパティが ECI にデフォルト値で定義されております。

編集する際は、直接 ECI を編集します。

表 14. コンテナプロパティ一覧

ステータスプロパティ Tag 名	概要
<b>SERVICE.EgSDAT.AutoRun</b>	サービス起動時の SDAT ファイル保存の自動開始を指定します。

### ■ サービス動作自動スタート指定 (.AutoRun)

Tag	備考	
Name	SERVICE.EgSDAT.AutoRun	サービス起動時に SDAT ファイルの保存を自動開始します。
Type	Boolean	
Size	1	
Value	TRUE (1)	初期値
値	意味	備考
FALSE (0)	サービス起動時には SDAT ファイル保存は行われません。	ユーザーアプリケーションからの EM_SERVICE_RUN 受信のタイミングで SDAT ファイル保存を開始します。
TRUE (1)	サービス起動時に SDAT ファイル保存を開始させる設定。	サービス起動時に自動的に SDAT ファイル保存を開始します。

## 7. 動作確認

### 7.1. 動作確認手順

RT-edge ソフトウェアを起動させ、SDAT サービスコンテナ設定が適切に反映されていることを確認します:

- 1) INtime カーネルを起動します。
- 2) RT-edge システムを開始します。(開始方法は RT-edge ユーザーズマニュアルをご参照ください。)
- 3) (RT-edge インストールパス)¥EgBrow.exe を起動します。



RT-edge 起動直後に RT-edge オブジェクトブラウザを起動すると初期化中の為、想定されるタグが表示されない場合があります。一度 RT-edge オブジェクトブラウザを終了し、再度起動させてください。

- 4) サービスコンフィグファイルで定義したタグが表示されていることを確認します。
- 5) 実行確認: SDAT ファイルが保存され、その結果がタグデータに反映されることを確認します。以下の図 24. タグデータ更新の確認では SDAT ファイルデータ終端を示すタグ「.AddPointer」が増加されることを確認しています。

Name	Current Value	Type	Source	Comment
SERVICE.EgSDAT	00000000 (0)	Int32		
SERVICE.EgSDAT.Status	01 (1)	byte		
SERVICE.EgSDAT.Error	False (0)	bool		
SERVICE.EgSDAT.Run	True (1)	bool		
SERVICE.EgSDAT.Live	000000b8 (184)	UInt32		
SERVICE.EgSDAT.MboxNameValue	00000006 (6)	UInt32		
SERVICE.EgSDAT.MboxColSize	00000800 (2048)	UInt32		
SERVICE.EgSDAT.MboxRowSize	00000400 (1024)	UInt32		
SERVICE.EgSDAT.AutoRun	True (1)	bool		TRUE=サービス起動時SDATファイル
SERVICE.EgSDAT.FragmentSize	0000000000000400 (1024)	Int64		SDATファイルサイズ(バイト)
SERVICE.EgSDAT.CapacitySize	0000000000001000 (4096)	Int64		SDATファイルサイズ(バイト)
SERVICE.EgSDAT.LastErr	00000000 (0)	Int32		
SERVICE.EgSDAT.FileFreeSize	00000000000001fb (507)	Int64		
SERVICE.EgSDAT.FormatNumber	00000001 (1)	Int32		
SERVICE.EgSDAT.AddPointer	00000217 (535)	UInt32		
SERVICE.EgSDAT.FileCount	00000004 (4)	Int32		
SERVICE.EgSDAT.CapacityFreeSize	0000000000000000 (0)	Int64		
SERVICE.EgSDAT.MeasureCount	00000000 (0)	Int32		
SERVICE.EgSDAT.TotalRecTime	0.0000000000000000	double		

図 24. タグデータ更新の確認

## 7.2. サンプルによる動作確認

---

サンプルとして ECI にデフォルトで EgTime サービスの現在時刻を定義しております。  
SDAT ファイルを開き、レコード構成、レコードバイナリが保存されていることを確認ください:

SDAT ファイル仕様については「2.3. SDAT ファイルフォーマット」を参照ください。



SDAT コンテナ起動中は SDAT ファイルをロックしているため、SDAT ファイルを参照する際は、SDAT コンテナを停止させた状態で参照ください。

## 8. サービスインジケータタグ

RT-edge 基本ソフトウェアは、標準的に各サービスコンテナの状態を示すタグをサービスインジケータタグとして RT-edge Object に展開します。これらは ECI に記述しなくても RT-edge 基本ソフトウェアが自動生成し、一部はフレームワークが自動処理します。

### 8.1. 一覧

表 15. サービスインジケータタグ一覧

ステータスインジケータ Tag 名	備考
SERVICE.EgSDAT.Status	現在のサービス起動状態を示します
SERVICE.EgSDAT.Error	現在のサービスエラー状態を示します
SERVICE.EgSDAT.Run	現在のデータリフレッシュ動作の状態を示します
SERVICE.EgSDAT.Live	サービスが健全であることを示すカウンタ

### 8.2. サービス起動状態ステータス(.Status)

Tag		備考
Name	SERVICE.EgSDAT.Status	現在のサービス起動状態を示します
Type	Byte	
Size	1	
Value	0	
値	意味	備考
0	サービスは起動していません	SDAT サービスコンテナ終了処理で設定されます。
1	サービスは起動されています	SDAT サービスコンテナ起動処理で設定されます。

### 8.3. サービス異常状態ステータス(.Error)

Tag		備考
Name	SERVICE.EgSDAT.Error	現在のサービスのエラー発生状態を示します
Type	Boolean	
Size	1	
Value	FALSE (0)	初期値
値	意味	備考
FALSE (0)	エラーはありません	
TRUE (1)	エラーを生じています	現バージョンでは起動エラー時コンテナを終了するため、この値に更新されることはありません。



## 8.4. サービス実行状態ステータス(.Run)

Tag		備考
Name	SERVICE.EgSDAT.Run	現在のタグリフレッシュ動作の状態を示します
Type	Boolean	
Size	1	
Value	FALSE (0)	初期値

値	意味	備考
FALSE (0)	タグリフレッシュは停止しています	タグリフレッシュを行わない状態。サービスメッセージ EM_SERVICE_PAUSE 受信後、タグリフレッシュトリガスレッド動作時に設定されます。
TRUE (1)	タグリフレッシュは活性化しています。	タグリフレッシュ可能な状態。サービスメッセージ EM_SERVICE_RUN 受信後、タグリフレッシュトリガスレッド動作時に設定されます。

## 8.5. サービス実行カウンタ(.Live)

Tag		備考
Name	SERVICE.EgSDAT.Live	サービスが健全であることを示すカウンタ
Type	UInt32	
Size	4	
Value	0	初期値

値	意味	備考
0~0xffffffff	指定周期の満了回数	サービスプロセスの健全性確認手段として用意されています。値が変化していることで健全であることを表します。

## 9. サービスプロパティタグ

SDAT コンテナが生成するタグについて説明します。

なおこれらのタグは基本的にユーザーからの編集はできません。

### 9.1. 一覧

表 16. サービスプロパティタグ一覧

ステータスプロパティ Tag 名	概要
SERVICE.EgSDAT.FragmentSize	SDAT ファイルサイズ(バイト単位)を表示します
SERVICE.EgSDAT.FileFreeSize	残りの SDAT ファイルサイズ(バイト単位) を表示します。
SERVICE.EgSDAT.FormatNumber	最新のレコード構成番号を表示します。
SERVICE.EgSDAT.AddPointer	SDAT ファイルのデータの終端を表示します。
SERVICE.EgSDAT.FileCount	収納バスに保存された SDAT ファイル数を表示します。
SERVICE.EgSDAT.CapacitySize	記録容量(バイト単位)を表示します。
SERVICE.EgSDAT.CapacityFreeSize	残りの記録容量(バイト単位)を表示します。
SERVICE.EgSDAT.MeasureCount	スループット計測回数
SERVICE.EgSDAT.TotalRecTime	スループット計測結果 (ミリ秒単位)

## 9.2. SDAT ファイルサイズ (.FragmentSize)

Tag		備考
Name	SERVICE.EgSDAT.FragmentSize	SDAT ファイルサイズです。(バイト単位)
Type	Int64	
Size	8	
Value	0x20000000	初期値
値	意味	備考
0~	SDAT ファイルサイズ(バイト単位) 1024 未満を指定した場合、1024 に変更します。	

```
<Tags>
<Tag Name="SERVICE.EgSDAT.AutoRun" Type="1" Comment="TRUE=サービス起動時SDATファイル保存開始" Value="TRUE"/>
<Tag Name="SERVICE.EgSDAT.FragmentSize" Type="8" Comment="SDATファイルサイズ(バイト)" Value="2048"/>
<Tag Name="SERVICE.EgSDAT.CapacitySize" Type="8" Comment="SDATファイル記録容量(バイト)" Value="32768"/>
</Tags>
```

図 25. ファイルサイズ(.FragmentSize)設定例

## 9.3. 残り SDAT ファイルサイズ (.FileFreeSize)

Tag		備考
Name	SERVICE.EgSDAT.FileFreeSize	残りの SDAT ファイルサイズです。(バイト単位)
Type	Int64	
Size	8	
Value	0x20000000	初期値
値	意味	備考
0~	残りの SDAT ファイルサイズで、レコード登録時に減少します。 残りのファイルサイズがレコードサイズ未満の場合は、次回レコード登録時、SDAT ファイルを新規作成します。	

## 9.4. 最新レコード構成番号 (.FormatNumber)

Tag		備考
Name	SERVICE.EgSDAT.FormatNumber	最新のレコード構成番号です。
Type	Int32	
Size	4	
Value	0	初期値
値	意味	備考
0～	次回レコード構成生成時に設定するレコード構成番号で、レコード構成生成時にインクリメントします。	

## 9.5. SDAT ファイルデータ終端 (.AddPointer)

Tag		備考
Name	SERVICE.EgSDAT.AddPointer	SDAT ファイルのデータ終端を示します。
Type	Int32	
Size	4	
Value	0	初期値
値	意味	備考
0～	レコード登録時、レコードサイズ数加算します。	

## 9.6. 記録容量 (.CapacitySize)

Tag		備考
Name	SERVICE.EgSDAT.CapacitySize	SDAT ファイルを収納する記録容量です。 (バイト単位)
Type	Int64	
Size	8	
Value	0x100000000	初期値

値	意味	備考
0~	記録容量(バイト単位) 「ファイルサイズ(.FragmentSize) × 2」未満を 指定した場合、「ファイルサイズ × 2」に変更 します。	

```
<Tags>
<Tag Name="SERVICE.EgSDAT.AutoRun" Type="1" Comment="TRUE=サービス起動時SDATファイル保存開始" Value="TRUE"/>
<Tag Name="SERVICE.EgSDAT.FragmentSize" Type="8" Comment="SDATファイルサイズ(バイト)" Value="2048"/>
<Tag Name="SERVICE.EgSDAT.CapacitySize" Type="8" Comment="SDATファイル記録容量(バイト)" Value="32768"/>
</Tags>
```

図 26. 記録容量(.CapacitySize)設定例

## 9.7. 残り記録容量 (.CapacityFreeSize)

Tag		備考
Name	SERVICE.EgSDAT.CapacityFreeSize	SDAT ファイルを収納する記録容量の残り サイズです。(バイト単位)
Type	Int64	
Size	8	
Value	0x100000000	初期値

値	意味	備考
0~	SDAT ファイルを収納する残りの記録容量で、 SDAT ファイル作成時に減少します。  残りの記録容量が SDAT ファイルサイズ未満の 場合は、次回 SDAT ファイル作成時、古い SDAT ファイルを削除します。	

## 9.8. スループット計測回数 (.MeasureCount)

Tag		備考
Name	SERVICE.EgSDAT.MeasureCount	SDAT コンテナのスループット計測回数です。
Type	Int32	
Size	4	
Value	0	初期値

値	意味	備考
0～	SDAT コンテナのスループット計測用に定義されたタグです。 ユーザーが指定した回数スループットを計測し、スループットの合計値を計測結果タグ.TotalRecTime に設定します。 スループット計測中は計測回数がデクリメントされ、0 になった時点で計測完了します。	スループット計測する際はこのタグ値を変更してください。



スループット計測は、「レコードバイナリ追加を受信直後」から「SDAT ファイルへ書き込み完了」までを計測します。  
計測回数が 0 以外の場合は、計測中です。

## 9.9. スループット計測結果 (.TotalRecTime)

Tag		備考
Name	SERVICE.EgSDAT.TotalRecTime	SDAT コンテナのスループット計測結果（ミリ秒単位）です。
Type	Double	
Size	8	
Value	0	初期値

値	意味	備考
0～	SDAT コンテナのスループット計測用に定義されたタグです。 ユーザーが指定した計測回数分のスループット計の合計値をミリ秒単位で設定します。	



このタグは前回値を保持しており、計測開始時に 0 リセットされません。  
計測開始時はこのタグ値を 0 リセットしてからご使用ください。

# 10. サービスメッセージ

SDAT サービスコンテナでは以下のメッセージに対する処理が実装されています:

表 17. サービスメッセージ一覧

メッセージ名	番号	説明
EM_SERVICE_STOP	101	サービスを終了させます。
EM_SERVICE_RUN	102	データ更新処理を開始します。
EM_SERVICE_PAUSE	103	データ更新処理を一時停止します。再開するためには、EM_SERVICE_RUN を送信してください。
EM_SERVICE_UPDATE	104	データ更新処理を指令します。データリフレッシュを 1 回実施します。 ※SDAT コンテナでは使用していません。
SDAT_CREATE_FORMAT_CMD	20000	内部動作に予約済みのメッセージです。
SDAT_INSERT_BINARY_CMD	20001	内部動作に予約済みのメッセージです。

各メッセージ送信後のサービスインジケータタグは以下の状態に遷移します:

表 18. サービスインジケータ状態

サービスインジケータタグ	初期化完了 (AutoRun=True)	初期化完了 (AutoRun=False)	EM_SERVICE_ STOP	EM_SERVICE_ RUN	EM_SERVICE_ PAUSE
SERVICE.EgSDAT.Status	1	1	0	1	1
SERVICE.EgSDAT.Run	TRUE	FALSE	FALSE	TRUE	TRUE
SERVICE.EgSDAT.Live	増加	停止	停止	増加	増加
SERVICE.EgSDAT.Error	FALSE	FALSE	FALSE	FALSE	FALSE

# 11. RT-edge タグデータの妥当性について

タグリンクされた RT-edge タグのデータは、サービスインジケータタグが以下の状態になっている時、妥当であると判断出来ます。

1. SERVICE.EgSDAT.Error = FALSE (エラーが発生していない)
2. SERVICE.EgSDAT.Run = TRUE (タグリフレッシュが実行されている)

上記の状態になっていない場合には、SDAT ファイル保存に何らかの問題が発生しています。詳細は、以下のトラブルシューティングをご参照ください。

「13.1. サービスインジケータを確認すると.Run が false になっています。」

## 12. SDAT ファイルへのアクセス

SDAT ファイルへのアクセスは以下の手順で行います：

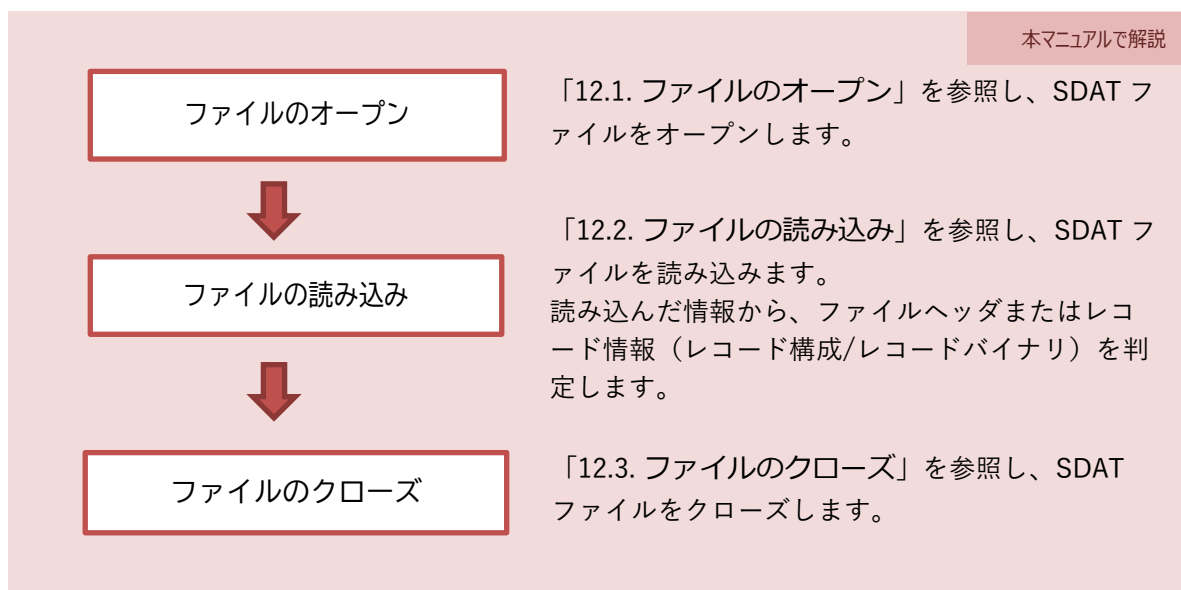


図 27.SDAT ファイルへのアクセス手順

以下の図 28. SDAT ファイルアクセスサンプルは C#コンソールアプリケーションのサンプルです。

```
/// <summary>
/// SDAT ファイル：タグ情報
/// -----
/// レコード構成のタグ情報格納用に作成したクラスです。
/// -----
/// </summary>
public class SDATTagFormat
{
    /// <summary>
    /// タグ名
    /// </summary>
    public string Name;
    /// <summary>
    /// タグタイプ
    /// </summary>
    public byte Type;
    /// <summary>
    /// タグサイズ
    /// </summary>
    public byte Size;
}

/// <summary>
/// SDAT ファイル：レコード構成
/// -----
/// レコード構成格納用に作成したクラスです。
/// -----
/// </summary>
public class SDATRecordFormat
{
    /// <summary>
    /// レコード構成番号
    /// </summary>
    public UInt16 FormatNo;
```



```
/// <summary>
/// レコードサイズ
/// </summary>
public UInt16 RecordSize;
/// <summary>
/// タグ数
/// </summary>
public UInt16 TagsCount;
/// <summary>
/// タグ情報：タグ数分必要のため配列で宣言します。
/// </summary>
public SDATTagFormat[] Tags;
}

/// <summary>
/// メインプログラム
/// -----
/// SDAT ファイル情報（レコードヘッダ、レコード構成、レコードバイナリ）をコンソール出力します。
/// レコード構成またはレコードバイナリの判定用に検出したレコード構成はリストに追加します。
/// -----
/// </summary>
/// <param name="args">起動引数：このプログラムでは未使用です。
static void Main(string[] args)
{
    Console.WriteLine("SDAT file open sample program");

    // コンソール出力用文字列
    string ConsoleMsg = "";

    // レコード構成リストを初期化します。
    List<SDATRecordFormat> RecordFormatList = new List<SDATRecordFormat>();

    // -----
    // ファイルのオープン
    // -----

    // SDAT ファイルをバイナリファイル読み込み用にオープンします。
    var fs = new FileStream("C:\\RT-edge\\20220622102535_SDAT.SDAT", FileMode.Open);
    var br = new BinaryReader(fs);

    // 読み込み位置を初期化します。
    UInt32 ReadPos = 0;

    // -----
    // ファイルヘッダの読み込み
    // ファイルヘッダは先頭 32 バイト固定のため、一括で読み込みます。
    // -----
    // ファイルヘッダ格納バッファを生成します。
    byte[] FileHeader = new byte[32];
    // ファイルヘッダを読み込みます。
    FileHeader = br.ReadBytes(FileHeader.Length);
    // 読み込み位置を更新します。
    ReadPos += (UInt32)FileHeader.Length;

    // SDAT ファイルのデータの終端を取得します。
    UInt32 EndOfData = BitConverter.ToUInt32(FileHeader, 9);

    // ファイルヘッダをコンソール出力します。
    ConsoleMsg = "FileHeader:";
    ConsoleMsg += " ID=" + Encoding.GetEncoding("Shift_JIS").GetString(FileHeader, 0, 4);
    ConsoleMsg += " Size=" + FileHeader[4].ToString();
    ConsoleMsg += " LatestRecord=" + BitConverter.ToUInt32(FileHeader, 5).ToString();
    ConsoleMsg += " EndOfData=" + EndOfData.ToString();
    Console.WriteLine(ConsoleMsg);

    // -----
    // レコードの読み込み
    // レコード読み込み位置がデータの終端まで行います。
    // -----
    while (true)
```

```
{  
  
    // -----  
    // レコードの読み込み  
    // -----  
  
    // レコードサイズを取得します。  
    UInt16 RecordSize = br.ReadUInt16();  
    // 読み込んだレコード情報を格納するためのバッファを確保します。  
    // バッファサイズはレコードサイズ(2 バイト)の除いたサイズです。  
    byte[] RecordInfo = new byte[RecordSize - 2];  
    // SDAT ファイルからレコード情報を読み込みます。  
    RecordInfo = br.ReadBytes(RecordInfo.Length);  
    // 読み込み位置を更新します。  
    ReadPos += RecordSize;  
  
    // レコード情報読み込み位置を初期化します。  
    int RecordPos = 0;  
  
    // レコード構成番号を取得します  
    UInt32 RecordNo = BitConverter.ToInt32(RecordInfo, 0);  
    // レコード情報読み込み位置を更新します。  
    RecordPos += 4;  
  
    // レコード構成/レコードバイナリを判定します。  
    // 1. 取得したレコード構成番号がレコード構成リストに未登録数の場合はレコード構成と判断します。  
    // 2. 上記以外の場合はレコードバイナリと判断します。  
  
    // レコード構成インデックスを-1(未登録)に初期化します。  
    int RecordFormatIndex = -1;  
    // レコード構成リストを検索し、  
    // 取得したレコード構成番号と一致したレコード構成リストのインデックスを取得します。  
    for (var i = 0; i < RecordFormatList.Count; i++)  
    {  
        // 取得したレコード構成番号かどうかを判定します。  
        if (RecordNo == RecordFormatList[i].FormatNo)  
        {  
            // レコード構成インデックスにレコード構成リストのインデックスを設定します。  
            RecordFormatIndex = i;  
            // 取得したレコード構成番号が一致したため、検索を終了します。  
            break;  
        }  
    }  
  
    // レコード構成/レコードバイナリを判定します。  
    if (RecordFormatIndex == -1)  
    {  
        // -----  
        // 未登録のため、レコード構成と判断します。  
        // -----  
  
        // タグ数を取得します。  
        UInt16 TagCount = BitConverter.ToUInt16(RecordInfo, RecordPos);  
  
        // レコード情報読み込み位置を更新します。  
        RecordPos += 2;  
  
        // レコード構成リストに追加するレコード構成を生成します。  
        var newRecordFormat = new SDATRecordFormat();  
  
        // レコード構成番号を設定します。  
        newRecordFormat.FormatNo = (UInt16)RecordNo;  
  
        // レコードサイズを設定します。  
        newRecordFormat.RecordSize = RecordSize;  
  
        // タグ数を設定します。  
        newRecordFormat.TagsCount = TagCount;  
    }  
}
```

```
// レコード構成情報をコンソール出力します。
ConsoleMsg = "RecordFormat:";
ConsoleMsg += " No=" + newRecordFormat.FormatNo.ToString();
ConsoleMsg += " Size=" + newRecordFormat.RecordSize.ToString();
ConsoleMsg += " Tags=" + newRecordFormat.TagsCount.ToString();
Console.WriteLine(ConsoleMsg);

// タグ数分タグ情報を確保します。
newRecordFormat.Tags = new SDATTagFormat[TagCount];

// 取得したタグ情報を追加します。
for (var TagNo = 0; TagNo < TagCount; TagNo++)
{
    // タグアイテム（配列に設定するタグ情報）を生成します。
    var TagItem = new SDATTagFormat();

    // タグ名長さを取得します。
    byte TagNameSize = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;

    // タグ名変換用バッファを確保します。
    byte[] TagName = new byte[TagNameSize];
    // タグ名変換用バッファにタグ名をコピーします。
    Array.Copy(RecordInfo, RecordPos, TagName, 0, TagNameSize);
    // タグ名変換用バッファから文字列に Shift_JIS コードで変換します。
    TagItem.Name = Encoding.GetEncoding("Shift_JIS").GetString(TagName);
    // レコード情報読み込み位置を更新します。
    RecordPos += TagNameSize;

    // タグタイプを取得します。
    TagItem.Type = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;

    // タグサイズを取得します。
    TagItem.Size = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;

    // 生成したタグアイテムをリストへ設定します。
    newRecordFormat.Tags[TagNo] = TagItem;

    // タグアイテムをコンソール出力します。
    ConsoleMsg = "- Tag[" + TagNo.ToString() + "]";
    ConsoleMsg += " Name=" + TagItem.Name;
    ConsoleMsg += " Type=" + TagItem.Type.ToString();
    ConsoleMsg += " Size=" + TagItem.Size.ToString();
    Console.WriteLine(ConsoleMsg);
}

// レコード構成リストに生成したレコード構成を追加します。
RecordFormatList.Add(newRecordFormat);
}
else
{
    // -----
    // 登録済みのため、レコードバイナリと判断します。
    // -----

    // レコードバイナリサイズ、レコード構成番号をコンソール出力します。
    ConsoleMsg = "RecordBinary:";
    ConsoleMsg += " No=" + RecordNo.ToString();
    ConsoleMsg += " Size=" + RecordSize.ToString();

    // タグデータを取得します。
    // タグデータの取得は、該当のレコード構成のタグ数を行います。
    for (var TagNo = 0; TagNo < RecordFormatList[(int)RecordNo].Tags.Count(); TagNo++)
    {
        // タグデータ格納用バッファを格納します。
    }
}
```

```

byte[] TagData = new byte[RecordFormatList[(int)RecordNo].Tags[TagNo].Size];
// タグデータを取得します。
Array.Copy(RecordInfo, RecordPos, TagData, 0, TagData.Count());
// レコード情報読み込み位置を更新します。
RecordPos += TagData.Length;

// コンソール出力用文字列に取得したタグデータを追加します。
ConsoleMsg += " 0x";
foreach (var TagDataElem in TagData.Reverse())
{
    ConsoleMsg += TagDataElem.ToString("X2");
}

// レコードバイナリ（タグデータ）をコンソール出力します。
Console.WriteLine(ConsoleMsg);
}

// -----
// ファイルの終端判定
// -----
// 読み込み位置がファイル終端の場合、読み込み処理を終了します。
if (ReadPos >= EndOfData) { break; }

}

// -----
// ファイルのクローズ
// -----
// ファイルオープンで使ったオブジェクトをクローズします。
br.Close();
fs.Close();

Console.WriteLine("exit program.");
}

```

図 28. SDAT ファイルアクセスサンプル

## 12.1. ファイルのオープン

実際に SDAT コンテナが出力した SDAT ファイルをオープンします。

### ■ ファイル形式

SDAT ファイルはバイナリ形式で保存されます。

SDAT ファイルをオープンする際は、バイナリ形式でオープンしてください。

### ■ ファイル名

SDAT ファイル名は「yyyyMMddHHmmss\_SDAT.SDT」で保存されます。

「yyyyMMddHHmmss」は SDAT コンテナのファイル生成日時を示します。



例えば、SDAT ファイル「20220620110340\_SDAT.SDT」は、SDAT コンテナが 2022 年 06 月 20 日 11 時 03 分 40 秒に生成したことを示します。

### ■ ファイルサイズ

SDAT ファイルサイズはデフォルト 512MB で、SDAT ファイル生成時に一括で確保するため、ファイルサイズの変動はありません。



ファイルサイズの変更は ECI ファイルにサービスプロパティタグ「.FragmentSize」を登録してください。  
なお、「.FragmentSize」に設定可能な最小値は 1024 です。

### ■ 保存先フォルダ

SDAT ファイルは SDAT コンテナのカレントパスに保存されます。



例えば、SDAT コンテナが「C:¥RT-edge¥」で起動されている場合、SDAT ファイルは「C:¥RT-edge¥yyyyMMddHHmmss\_SDAT.SDT」で保存されます。

### ■ ファイル保存数

SDAT コンテナはデフォルト最大 4GB の記録容量で SDAT ファイルを管理します。

ファイルサイズが 512MB の場合、SDAT コンテナが生成する最大ファイル数は  $4\text{GB} / 512\text{MB} = 8$  ファイルです。



記録容量の変更は ECI ファイルにサービスプロパティタグ「.CapacitySize」を登録してください。  
なお、「.CapacitySize」に設定可能な最小値は「ファイルサイズ (.Fragment) × 2」です。

## ■ ファイルヘッダ

SDAT ファイルの先頭 32 バイトにファイルヘッダが保存されます。  
ファイルヘッダは以下の表 19. ファイルヘッダ内容で構成されます。

表 19. ファイルヘッダ内容

オフセット	項目	内容
0x00000000 : 0x00000003	SDAT ファイル識別子	SDAT ファイル識別子"SDAT" が Shift_JIS コードで格納されます。
0x00000004	ファイルヘッダサイズ	ファイルヘッダのバイトサイズ=32 が格納されます。
0x00000005 : 0x00000008	最新レコード構成番号	最新のレコード構成番号が格納されます。 レコード構成登録時、インクリメントされます。
0x00000009 : 0x0000000C	データの終端	レコードの収納最終位置です。 レコード登録時、レコードサイズ分インクリメントされます。
0x0000000D : 0x0000001F	(予備)	将来用に確保したエリアです。

以下の図 29. ファイルヘッダ格納例は、SDAT ファイルに格納されたファイルヘッダをバイナリエディタで開いた図です。

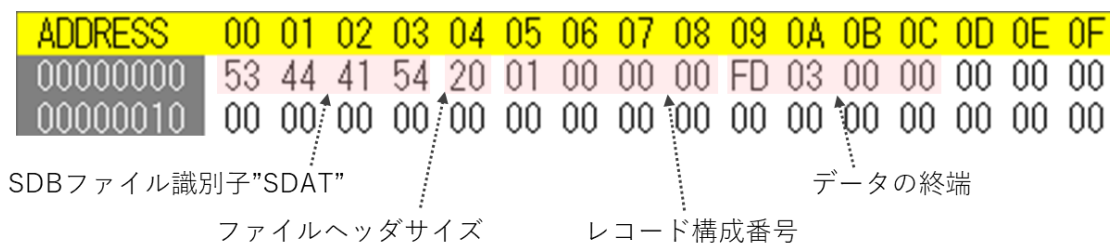


図 29. ファイルヘッダ格納例

表 20. ファイルヘッダ内容

オフセット	項目	値
0x00000000～0x00000003	SDAT ファイル識別子	"SDAT"
0x00000004	ファイルヘッダサイズ	0x20
0x00000005～0x00000008	レコード構成番号	0x00000008
0x00000009～0x0000000C	データの終端	0x00081C72

## ■ レコード情報

SDAT ファイルの 32 バイト目以降にタグ情報を収納した「レコード構成」、タグデータを収納した「レコードバイナリ」が保存されます。

### レコード構成

表 21. レコード構成内容

オフセット	項目	内容
0x00000000	レコード構成サイズ	レコード構成サイズがバイト単位で格納されます。
:		
0x00000001		
0x00000002	レコード構成番号	レコード構成.レコード構成番号が格納されます。
:		
0x00000005		
0x00000006	タグ数	レコード構成.タグ数 (1~n) が格納されます。
:		
0x00000007		
0x00000008	タグ 1.タグ名長さ	レコード構成.タグ 1 の長さ (NULL は除く) がバイト単位で格納されます。
0x00000009	タグ 1.タグ名	レコード構成.タグ 1 のタグ名 (NULL は除く Shift_JIS コード) が格納されます。
:		
:	タグ 1.タグタイプ	レコード構成.タグ 1 のタグタイプが格納されます。タグタイプについては以下「表 22. タグタイプ(型)、タグサイズ」をご参照ください。
:	タグ 1.タグサイズ	レコード構成.タグ 1 のタグサイズが格納されます。タグサイズについては「表 22. タグタイプ(型)、タグサイズ」をご参照ください。
:		レコード構成.タグ 2 からタグ n-1 までのタグ情報 (タグ長さ、タグ名、タグタイプ、タグサイズ) が収納されます。
:	タグ n.タグ名長さ	レコード構成.タグ n の長さ (NULL は除く) がバイト単位で格納されます。
:	タグ n.タグ名	レコード構成.タグ n のタグ名 (NULL は除く Shift_JIS コード) が格納されます。
:	タグ n.タグタイプ	レコード構成.タグ n のタグタイプが格納されます。
:	タグ n.タグサイズ	レコード構成.タグ n のタグサイズが格納されます。

表 22. タグタイプ(型)、タグサイズ

No	型	サイズ(byte)	用途
0	UNDEFINE	-	未定義
1	Boolean	1	bool 値
2	SByte	1	符号付き 8 ビット整数
3	Byte	1	符号なし 8 ビット整数
4	Int16	2	符号付き 16bit 整数
5	UInt16	2	符号なし 16bit 整数
6	Int32	4	符号付き 32bit 整数
7	UInt32	4	符号なし 32bit 整数
8	Int64	8	符号付き 64bit 整数
9	UInt64	8	符号なし 64bit 整数
10	Float	4	単精度実数(32bit)
11	Double	8	倍精度実数(64bit)

以下の図 30. レコード構成格納例は、SDAT ファイルに格納されたレコード構成をバイナリエディタで開いた図です。

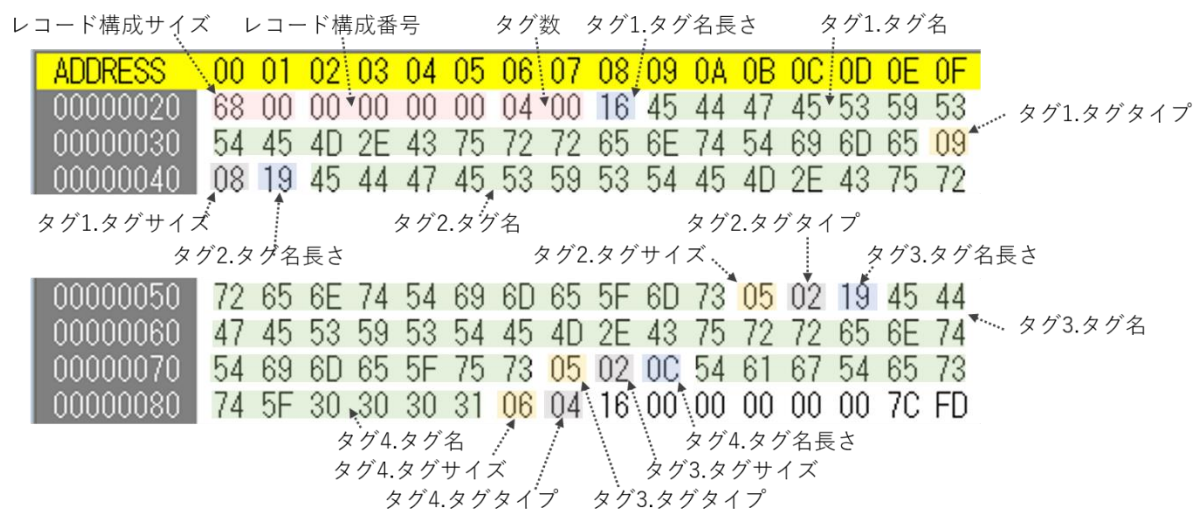


図 30. レコード構成格納例

表 23.レコード構成格納例

オフセット	項目	値
0x00000000~0x00000001	レコード構成サイズ	0x0068
0x00000002~0x00000005	レコード構成番号	0x00000000
0x00000006~0x00000007	タグ数	0x0004
0x00000008	タグ 1.タグ名長さ	0x16
0x00000009~0x0000001E	タグ 1.タグ名	"EDGESYSTEM.CurrentTime"
0x0000001F	タグ 1.タグタイプ	0x09
0x00000020	タグ 1.タグサイズ	0x08
0x00000021	タグ 2.タグ名長さ	0x19
0x00000022~0x0000003A	タグ 2.タグ名	"EDGESYSTEM.CurrentTime_ms"
0x0000003B	タグ 2.タグタイプ	0x05
0x0000003C	タグ 2.タグサイズ	0x02
0x0000003D	タグ 3.タグ名長さ	0x19



オフセット	項目	値
0x0000003E～0x00000056	タグ 3.タグ名	"EDGESYSTEM.CurrentTime_us"
0x00000057	タグ 3.タグタイプ	0x05
0x00000058	タグ 3.タグサイズ	0x02
0x00000059	タグ 4.タグ名長さ	0xC
0x0000005A～0x00000065	タグ 4.タグ名	"TagTest_0001"
0x00000066	タグ 4.タグタイプ	0x06
0x00000067	タグ 4.タグサイズ	0x04

## レコードバイナリ

表 24. レコードバイナリ内容

オフセット	項目	内容
0x00000000 : 0x00000001	レコードバイナリサイズ	レコードバイナリサイズがバイト単位で格納されます。
0x00000002 : 0x00000005	レコード構成番号	レコードバイナリが属するレコード構成のレコード構成番号が格納されます。
0x00000006 : :	タグ 1.データ	レコード構成番号に指定されたレコード構成.タグ 1 のデータが格納されます。
:		レコード構成番号に指定されたレコード構成.タグ 2～n-1 のデータが格納されます。
:      タグ n.データ		レコード構成番号に指定されたレコード構成.タグ n のデータが格納されます。

以下の図 31. レコードバイナリ格納例は、SDAT ファイルに格納されたレコードバイナリをバイナリエディタで開いた図です。

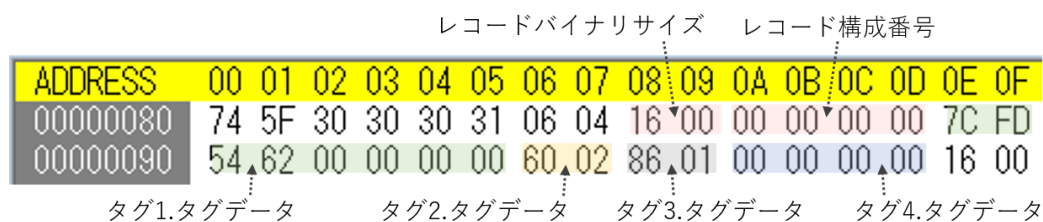


図 31. レコードバイナリ格納例

表 25.レコードバイナリ格納例

オフセット	項目	値
0x00000000～0x00000001	レコードバイナリサイズ	0x0016
0x00000002～0x00000005	レコード構成番号	0x00000000
0x00000006～0x0000000D	タグ 1.タグデータ	0x00006254FD7C
0x0000000E～0x0000000F	タグ 2.タグデータ	0x0260
0x00000010～0x00000011	タグ 3.タグデータ	0x0186
0x00000012～0x00000015	タグ 4 タグデータ	0x00000000

### ■ ファイル切替

SDAT ファイルの切替は、「レコード書き込み時のレコードサイズ(ファイル終端(プロパティタグ".AddPointer") + レコードバイナリサイズ)」と「残り SDAT ファイルサイズ(プロパティタグ".FileFreeSize")」で比較します。

表 26. ファイル切替条件

条件	ファイル切替
レコード書き込み時のレコードサイズ ≤ 残り SDAT ファイルサイズ	なし
レコード書き込み時のレコードサイズ > 残り SDAT ファイルサイズ	あり

### ■ ファイル削除

SDAT コンテナは、新規ファイル生成時に保存先フォルダの残りの記録容量をチェックします。

残りの記録容量がファイルサイズ未満の場合は、一番古い作成日時にファイルを削除し、新規ファイルを生成します。

以下の図 32. ファイル切替は、ファイル切替時に 20240201010203\_SDAT.SDT が削除され、20240201010203\_SDAT.SDT が作成されます。

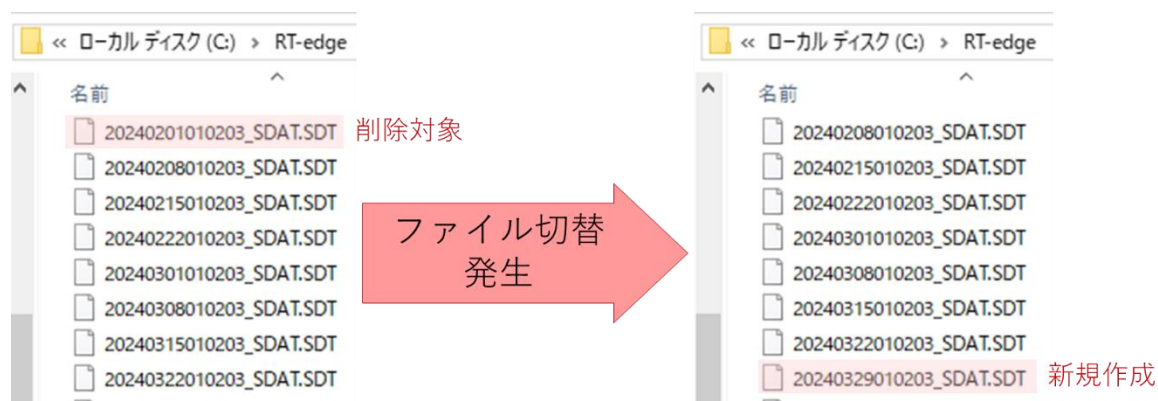


図 32. ファイル切替

### ■ ファイルオープン時の注意事項

SDAT コンテナ起動中は出力中の SDAT ファイルをロックしているため、出力中の SDAT ファイルをオープンすることはできません。SDAT コンテナ停止後、ファイルをオープンしてください。

もし出力中の SDAT ファイルをオープンしたい場合は、出力中の SDAT ファイルのコピーをオープンしてください。また SDAT コンテナがオープンしていない SDAT ファイルについては、SDAT コンテナはロックしていないため、オープン可能です。

### ■ ロック中のファイルかどうかの見分け方

SDAT コンテナは、ファイル保存先フォルダにある最新の作成日時のファイルに書き込みます。以下の図 33. ファイルロックは、20240322010203\_SDAT.SDT がロック中のファイルです。

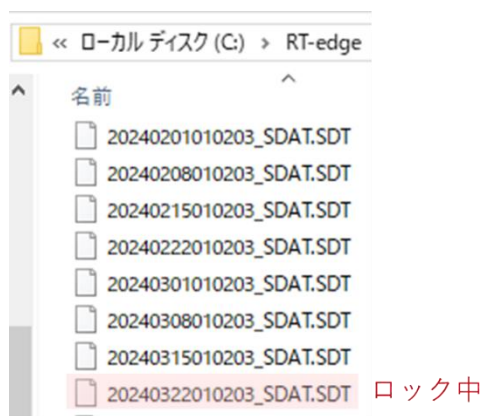


図 33. ファイルロック

### ■ ファイルのオープン

SDAT ファイルはバイナリファイルのため、バイナリファイルでオープンします。

```
// SDAT ファイルをバイナリファイル読み込み用にオープンします。
var fs = new FileStream("C:\\RT-edge\\20220622102535_SDAT.SDT", FileMode.Open);
var br = new BinaryReader(fs);
```

図 34. ファイルオープン

## 12.2. ファイルの読み込み

以下の図 35. ファイル読み込み手順でオープン中の SDAT ファイルの読み込みを行います。

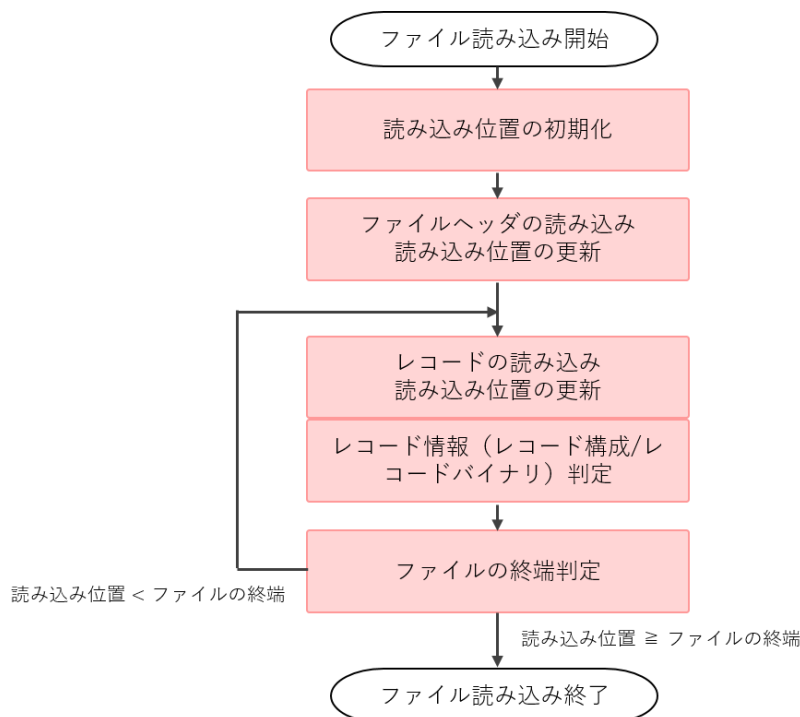


図 35. ファイル読み込み手順

### ■ 読み込み位置の初期化

SDAT ファイルの読み込みは、読み込み位置（初期値=0）を読み込んだサイズ数インクリメントしながら行います。

```
// 読み込み位置を初期化します。
UInt32 ReadPos = 0;
```

図 36. 読み込み位置の初期化

### ■ ファイルヘッダの読み込み

SDAT ファイルのファイルヘッダは、ファイルの先頭 32 バイトに格納されております。ファイルヘッダについては、表 19. ファイルヘッダ内容をご参照ください。

以下の手順で、SDAT ファイルからファイルヘッダを読み込み、ファイルヘッダに格納された各種情報を取得します。

- 1) ファイルヘッダの読み込み :  
SDAT ファイルから先頭 32 バイトを読み込みます。
- 2) 読み込み位置の更新 :  
読み込み位置をファイルヘッダサイズ（32 バイト）インクリメントします。
- 3) ファイル終端の取得 :  
読み込んだファイルヘッダからファイルクローズ判定用にファイル終端を取得します。

```
// ファイルヘッダ格納バッファを生成します。
```

```
byte[] FileHeader = new byte[32];
// ファイルヘッダを読み込みます。
FileHeader = br.ReadBytes(FileHeader.Length);
// 読み込み位置を更新します。
ReadPos += (UInt32)FileHeader.Length;

// SDAT ファイルのデータの終端を取得します。
UInt32 EndOfData = BitConverter.ToUInt32(FileHeader, 9)
```

図 37. ファイルヘッダの読み込み

## ■ レコードの読み込み

レコードの読み込みは、以下の手順で読み込みます。

- 1) レコードサイズの読み込み :  
SDAT ファイルからレコード構成/レコードバイナリのレコードサイズを読み込みます。
- 2) レコードの読み込み :  
SDAT ファイルからレコード構成/レコードバイナリを読み込みます。  
読み込むサイズはレコードサイズ (2 バイト) を除いたサイズです。
- 3) 読み込み位置の更新 :  
読み込み位置をレコードサイズインクリメントします。

```
// レコードサイズを取得します。
UInt16 RecordSize = br.ReadUInt16();
// 読み込んだレコード情報を格納するためのバッファを確保します。
// バッファサイズはレコードサイズ(2 バイト)の除いたサイズです。
byte[] RecordInfo = new byte[RecordSize - 2];
// SDAT ファイルからレコード情報を読み込みます。
RecordInfo = br.ReadBytes(RecordInfo.Length);
// 読み込み位置を更新します。
ReadPos += RecordSize;
```

図 38. レコードの読み込み

## ■ レコード情報 (レコード構成/レコードバイナリ) 判定

読み込んだレコード情報からレコード構成番号を取得します。

取得したレコード構成番号からレコード構成またはレコードバイナリを判定します。

表 27. レコード情報判定

判定条件	レコード情報
取得したレコード構成番号が今までに取得していないレコード構成番号	レコード構成
取得したレコード構成番号が取得済みのレコード構成番号	レコードバイナリ

下の図 39. レコード情報 (レコード構成/レコードバイナリ) 判定では、取得したレコード構成番号とリストに登録されたレコード構成番号を比較し、レコード構成番号がリストに見つからない場合は、レコード構成と判定する例です。

```
// レコード情報読み込み位置を初期化します。
int RecordPos = 0;

// レコード構成番号を取得します
UInt32 RecordNo = BitConverter.ToUInt32(RecordInfo, 0);
// レコード情報読み込み位置を更新します。
RecordPos += 4;
```

```
// レコード構成/レコードバイナリを判定します。
// 1. 取得したレコード構成番号がレコード構成リストに未登録数の場合はレコード構成と判断します。
// 2. 上記以外の場合はレコードバイナリと判断します。

// レコード構成インデックスを-1(未登録)に初期化します。
int RecordFormatIndex = -1;
// レコード構成リストを検索し、
// 取得したレコード構成番号と一致したレコード構成リストのインデックスを取得します。
for (var i = 0; i < RecordFormatList.Count; i++)
{
    // 取得したレコード構成番号かどうかを判定します。
    if (RecordNo == RecordFormatList[i].FormatNo)
    {
        // レコード構成インデックスにレコード構成リストのインデックスを設定します。
        RecordFormatIndex = i;
        // 取得したレコード構成番号が一致したため、検索を終了します。
        break;
    }
}

// レコード構成/レコードバイナリを判定します。
if (RecordFormatIndex == -1)
{
    // -----
    // 未登録のため、レコード構成と判断します。
    // -----
}
else
{
    // -----
    // 登録済みのため、レコードバイナリと判断します。
    // -----
}
```

図 39. レコード情報（レコード構成/レコードバイナリ）判定

## レコード構成

読み込んだレコード構成（レコードサイズを除く）から、レコード情報読み込み位置を更新しながら、タグ数、タグ情報を取得します。レコード構成については、表 21. レコード構成内容をご参照ください。

レコード情報読み込み位置は、レコード読み込みで使用した読み込み位置とは別で、既にレコード構成番号取得後に更新しているため、レコード情報読み込み位置=4 から開始します。

### 1) タグ数の取得：

読み込んだレコード構成のレコード情報読み込み位置からタグ数を取得します。  
取得後、レコード情報読み込み位置をタグ数（2 バイト）インクリメントします。

### 2) タグ情報の取得：

タグ情報の取得はタグ数分行います。

### 3) タグ名の長さの取得：

読み込んだレコード構成のレコード情報読み込み位置からタグ名の長さを取得します。  
取得後、レコード情報読み込み位置をタグ名の長さ（1 バイト）インクリメントします。

- 4) タグ名の取得 :  
読み込んだレコード構成のレコード情報読み込み位置からタグ名を取得します。  
タグ名を取得する際の文字コードは、Shift\_JIS コードを指定します。  
取得後、レコード情報読み込み位置をタグ名（タグ名の長さ）インクリメントします。
- 5) タグタイプの取得 :  
読み込んだレコード構成のレコード情報読み込み位置からタグタイプを取得します。  
取得後、レコード情報読み込み位置をタグタイプ（2 バイト）インクリメントします。
- 6) タグサイズの取得 :  
読み込んだレコード構成のレコード情報読み込み位置からタグサイズを取得します。  
取得後、レコード情報読み込み位置をタグサイズ（2 バイト）インクリメントします。

```
// タグ数を取得します。
UInt16 TagCount = BitConverter.ToUInt16(RecordInfo, RecordPos);

// レコード情報読み込み位置を更新します。
RecordPos += 2;

// レコード構成リストに追加するレコード構成を生成します。
var newRecordFormat = new SDATRecordFormat();

// レコード構成番号を設定します。
newRecordFormat.FormatNo = (UInt16)RecordNo;

// レコードサイズを設定します。
newRecordFormat.RecordSize = RecordSize;

// タグ数を設定します。
newRecordFormat.TagsCount = TagCount;

// タグ数分タグ情報を確保します。
newRecordFormat.Tags = new SDATTagFormat[TagCount];

// 取得したタグ情報を追加します。
for (var TagNo = 0; TagNo < TagCount; TagNo++)
{
    // タグアイテム（配列に設定するタグ情報）を生成します。
    var TagItem = new SDATTagFormat();

    // タグ名長さを取得します。
    byte TagNameSize = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;

    // タグ名変換用バッファを確保します。
    byte[] TagName = new byte[TagNameSize];
    // タグ名変換用バッファにタグ名をコピーします。
    Array.Copy(RecordInfo, RecordPos, TagName, 0, TagNameSize);
    // タグ名変換用バッファから文字列に Shift_JIS コードで変換します。
    TagItem.Name = Encoding.GetEncoding("Shift_JIS").GetString(TagName);
    // レコード情報読み込み位置を更新します。
    RecordPos += TagNameSize;

    // タグタイプを取得します。
    TagItem.Type = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;

    // タグサイズを取得します。
    TagItem.Size = RecordInfo[RecordPos];
    // レコード情報読み込み位置を更新します。
    RecordPos++;
}
```

```
// 生成したタグアイテムをリストへ設定します。
newRecordFormat.Tags[TagNo] = TagItem;
}
```

図 40. レコード構成の取得

## レコードバイナリ

読み込んだレコードバイナリ（レコードサイズを除く）から、レコード情報読み込み位置を更新（タグサイズインクリメント）しながら、タグデータを取得します。レコードバイナリについては、表 24. レコードバイナリ内容をご参照ください。

レコード情報読み込み位置は、レコード読み込みで使った読み込み位置とは別で、既にレコード構成番号取得後に更新しているため、レコード情報読み込み位置=4 から開始します。

またタグデータを取得する際に使用するタグサイズは既に登録済みであることが前提です。

```
// タグデータを取得します。
// タグデータの取得は、該当のレコード構成のタグ数を行います。
for (var TagNo = 0; TagNo < RecordFormatList[(int)RecordNo].Tags.Count(); TagNo++)
{
    // タグデータ格納用バッファを格納します。
    byte[] TagData = new byte[RecordFormatList[(int)RecordNo].Tags[TagNo].Size];
    // タグデータを取得します。
    Array.Copy(RecordInfo, RecordPos, TagData, 0, TagData.Count());
    // レコード情報読み込み位置を更新します。
    RecordPos += TagData.Length;
}
```

図 41. レコードバイナリの取得

### ■ ファイルの終端判定

読み込み位置がファイルヘッダで取得したファイル終端であるかを判定します。

読み込み位置が終端である場合、レコードの読み込みを終了し、ファイルをクローズします。

読み込み位置が終端でない場合、引き続きレコードの読み込みを行います。

```
// 読み込み位置がファイル終端の場合、読み込み処理を終了します。
if (ReadPos >= EndOfData) { break; }
```

図 42. ファイルの終端判定

## 12.3. ファイルのクローズ

オープン中のファイルをクローズします。

```
// ファイルオープンで使ったオブジェクトをクローズします。
br.Close();
fs.Close();
```

図 43. ファイルクローズ



## 13. トラブルシューティング

### 13.1. サービスインジケータを確認すると.Run が false になっています。

- 原因：サービス開始要求を受け付けられていません。

サービス開始要求が無い、初期化処理が終わらずサービス開始要求を受け付けられていません。

- 対応：サービス開始要求を行います。

アプリケーションから「EM\_SERVICE\_RUN」メッセージを送る、または EgSDAT.xml にて「.AutoRun」プロパティの Value を「TRUE」に設定します。

### 13.2. RT-edge オブジェクトブラウザにサービスインジケータが表示されません。

- 原因：SDAT サービスコンテナ初期化処理中です。

SDAT サービスコンテナ初期化処理中では、タグが生成されていません。

- 対応：終了させ、10 秒程度待ったのち再度 RT-edge オブジェクトブラウザを起動させます。

SDAT サービスコンテナ初期化処理完了後に再度起動させます。（初期化処理でエラーになった場合はサービスインジケータの Error プロパティが true の状態で生成されます。）

### 13.3. SDAT ファイルにタグデータが保存されません。

- 原因：タグ設定がされていません。

サービスインジケータ「.Run」TRUE にも関わらず SDAT ファイルにタグデータが保存されない場合、タグ設定がされていない可能性があります。

- 対応：タグ設定を行います。

設定方法については「6.1. RT-edge Object 設定」を参照ください。



## 更新履歴

版	日付	更新説明
1	2021.12	初回版
2	2022.07	1) コンテナ名を「SDB」から「SDAT」へ変更 2) 動作環境の変更 3) 「SDAT ファイルへのアクセス」を追加
3	2024.03	1) コンポーネント名を「EgSDB」から「EgSDAT」変更 2) 「EgSDAT」変更に伴い、各種タグ名も変更 3) 「EgSDAT」変更の注意事項を追記
4	2025.08	RTCD の名称を「RT-edge Object」に変更

INDUSTRIAL REALTIME EDGE COMPUTERS

# SDAT Container ユーザーズマニュアル

発行元：株式会社マイクロネット

TEL: +81(0)299-90-1733

FAX: +81(0)299-92-8557

- ・本書の内容、及び付属のソフトウェアの全部または一部を無断で転載することは禁止しております。
- ・本製品の内容については、将来予告なしに変更することがあります。
- ・本製品の内容について万が一不審な点や記載間違いなどお気づきの点がございましたら、お手数ですが、当社までご連絡ください。
- ・Windows XP、Windows 7、Windows 8、Windows 10 等、Windows は、米国 Microsoft Corporation における登録商標です。
- ・Visual Studio、Visual C++等は、米国、およびその他の国における Microsoft Corporation の登録商標です。
- ・INtime は米国 TenAsys における登録商標です。
- ・その他、記載されている会社名、製品名は、各社の商標又は登録商標です