App Bridge Tools
通知連携サービスガイド

Ver.1.19.0

目次

１．本書の役割

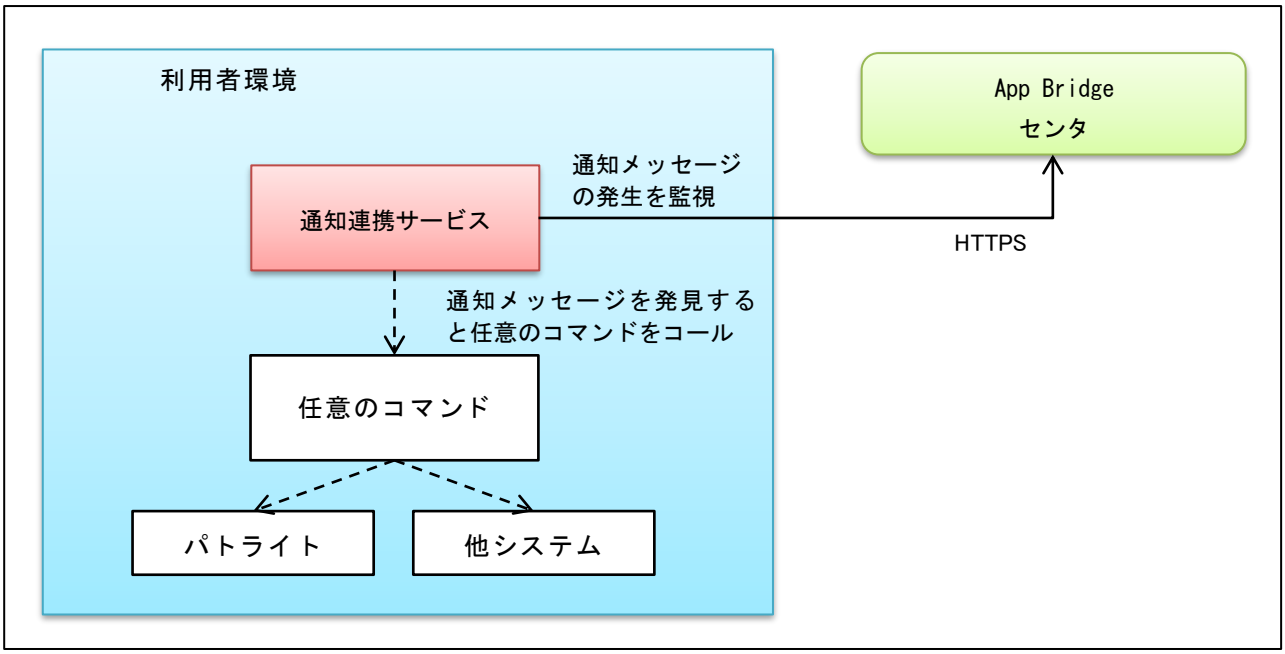
本書は、App Bridge が提供する通知連携サービスについて説明するものです。

２．通知連携サービスの概要

2.1 通知連携サービスとは

通知連携サービスは、App Bridge Monitor が提供する REST API を使用し、App Bridge センタで作成された通知メッセージを任意のアプリケーションに連携する Windows サービスです。

利用者の Windows サーバに通知連携サービスをインストールいただきます。インストールされた通知連携サービスは、App Bridge センタの REST API を使用して通知メッセージを監視し、新しい通知メッセージを発見すると、指定されたコマンド（プログラムやスクリプト）を起動します。これにより利用者環境でのパトライト点灯や任意のアプリケーション連携が可能となります。



2.2 監視対象となるオーナ

通知連携サービスを利用するためには、API を許可されたユーザアカウントが必要です。通知連携サービスは、設定されたユーザアカウントの権限を利用し、当該ユーザアカウントに許可されたアプリケーションオーナの通知メッセージを監視します。

2.3 通知連携サービスの稼働環境

通知連携サービスは以下の環境で稼働します。App Bridge Monitor Agent は必要ありませんが、同居は可能です。

項目	内容	備考
対象 OS	Windows Server 2012 Windows Server 2012 R2 Windows Server 2016 Windows Server 2019	
前提フレームワーク	.NET Framework 4.5.2 以降	
前提ネットワーク	App Bridge センタとの HTTPS 通信	プロキシサーバに対応しています

３．通知連携サービスのインストールと設定

3.1 セットアップファイルのダウンロードと解凍

対象のセットアップファイルを以下のサイトからダウンロードし、解凍します。

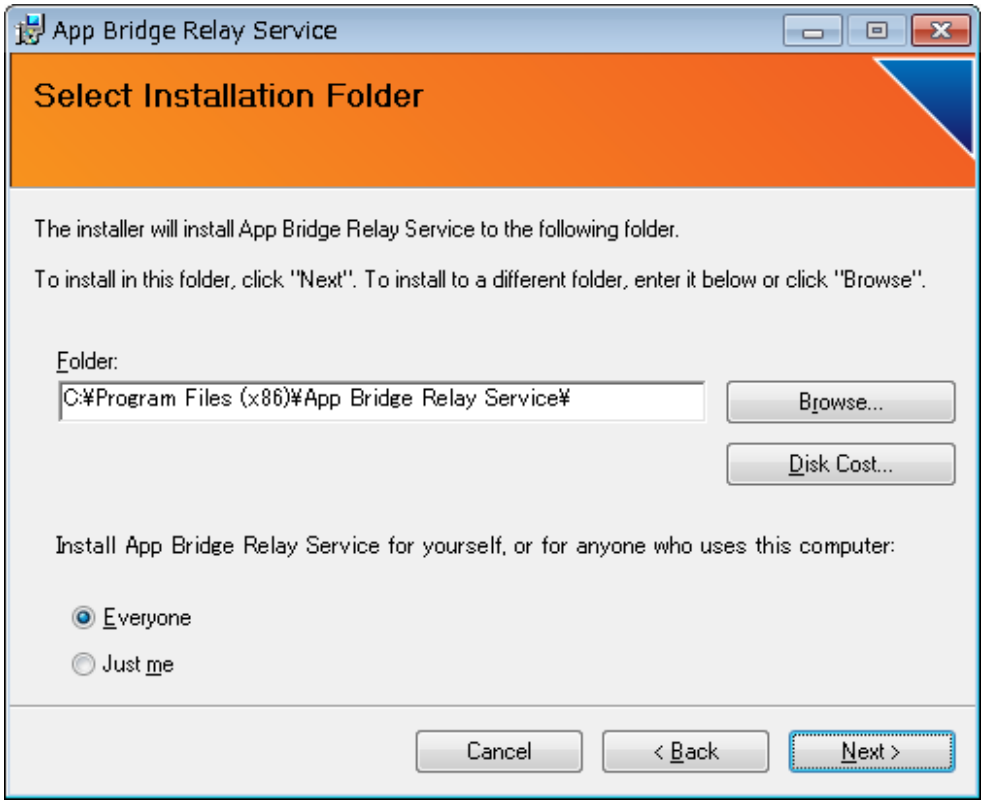
http://www.app-bridge.com/Setup_Tools.html

ダウンロードファイル	含まれるセットアップファイル	備考
SetupABMRelay_v011900.zip	SetupUCSRelaySRV.msi	
SetupABMAPIConfig_v011900.zip	SetupABMAPIConfig.msi	※1
SetupABMSupport_v011900.zip	SetupABMSupport.msi	※1、※2

※1：同セットアップが既にインストールされているとき、不要です
※2：App Bridge Monitor Windows Agent がインストールされているとき、不要です

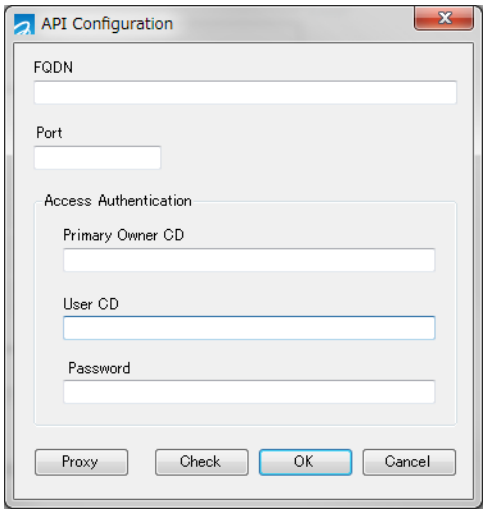
3.2 インストール

解凍したセットアップを実行し、示されたウィザードに従って、インストールを実施します。



3.3 ユーザアカウント設定

インストールしたサーバのメニューから、[スタート]→[App Bridge API]→[API Configuration]を選択し、表示された画面で通知連携サービスが使用するユーザアカウント情報を設定します。



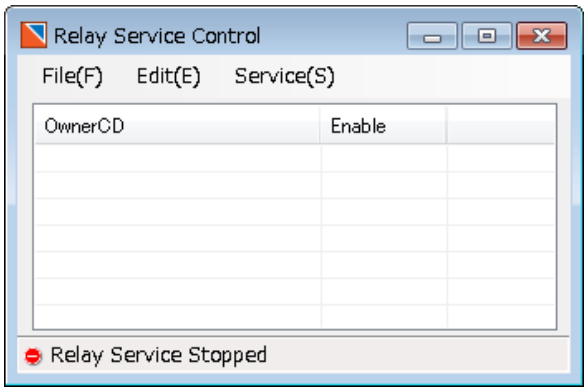
#	項目	内容	備考
1	FQDN	App Bridge センタにアクセスするための FQDN	mercury-api.app-bridge.com を指定下さい
2	Port	App Bridge センタにアクセスするためのポート番号	443 を指定下さい
3	Primary Owner CD	ユーザが所属するオーナコード	
4	User CD	アクセスに使用するユーザコード	
5	Password	ユーザのパスワード	
6	Proxy	プロキシサーバを経由する場合、本ボタンをクリックしプロキシ情報を設定します。	※1
7	Check	App Bridge センタと通信し、通信状態を確認します	

※1：プロキシ設定の詳細は、App Bridge Monitor Agent 監視サービス Windows Agent インストールガイド/プロキシの設定」をご参照下さい。

3.4 対象オーナと起動コマンドの設定

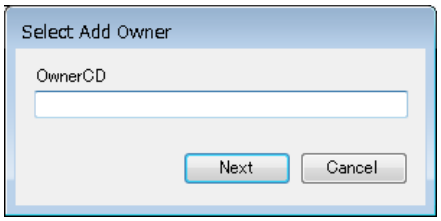
3.4.1 ツールの起動

通知連携サービスをインストールしたサーバのメニューから、[スタート]→[App Bridge Relay Service]→[Relay Control]を選択し、登録ツールを起動します。



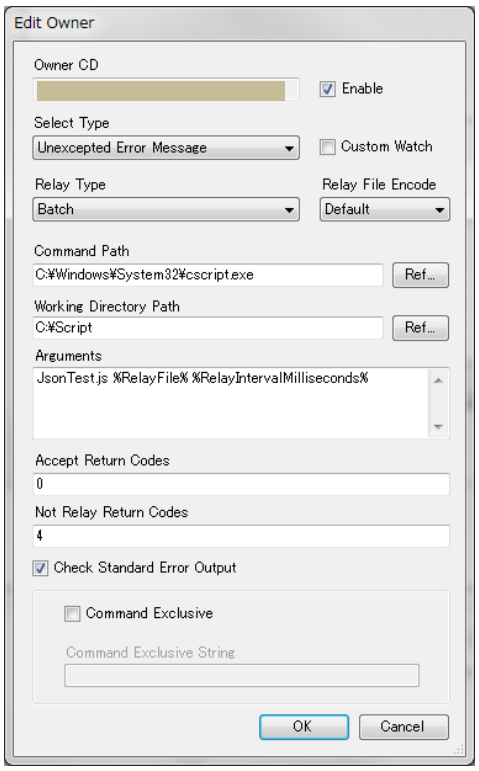
3.4.2 対象オーナの指定

ツールメニューの[Edit]→[Add Owner]を選択し、表示された画面で対象とするオーナコードを入力し、[Next]をクリックします。[Next]ボタンをクリックすると、App Bridge センタと通信し、入力されたオーナコードを検証します。



3.4.3 抽出条件と起動コマンドの設定

表示された画面に抽出する通知メッセージの条件、起動するコマンド情報を指定します。



項目	内容	備考
Enable	オーナを処理対象とする場合、チェックします	
Select Type	連携対象とする通知メッセージを選択します ● Unconfirmed Error Messages：未確認のエラーメッセージ ● Unexcepted Error Messages：通知抑止されていないエラーメッセージ ● All Error Messages：全てのエラーメッセージ ● All Messages：全てのメッセージ	
Custom Watch	カスタム監視のメッセージを連携対象とする場合チェックします	
Relay Type	以下からコマンド起動方式を選択します ● All Messages：通知メッセージ毎にコマンド起動 ● First Message：先頭の通知メッセージで起動 ● Batch：通知連携ファイルを作成し起動	
Relay File Encode	通知連携ファイルの文字コードを選択します。 Default は OS のデフォルト文字コードを意味します。 OS が日本語版の場合、シフト JIS となります	Relay Type：Batch のみ設定可能
Command Path	起動するコマンドパスを指定します	
Working Directory Path	コマンドの属するフォルダと異なるフォルダを Working Directory とする場合に指定します	
Arguments	コマンドに引き渡す引数を指定します	（※1）
Accept Return Codes	正常終了した場合のリターンコードを指定します	（※2）
Not Relay Return Codes	正常終了したが、連携処理を実施しなかった場合のリターンコードを指定します	（※2）、（※3）
Check Standard Error	標準エラー出力に出力があったとき、異常とみなす場	

Output	合にチェックします	
Command Exclusive	コマンドを排他実行する場合にチェックします	
Command Exclusive String	Command Exclusive をチェックした場合、排他制御に使用する文字列（英字、数字、_（アンダスコア）からなる 1～100 文字）を指定します	同じ Command Exclusive String を指定されたコマンドが排他制御されます

※1：詳細は後述の「コマンド起動方式」をご参照下さい

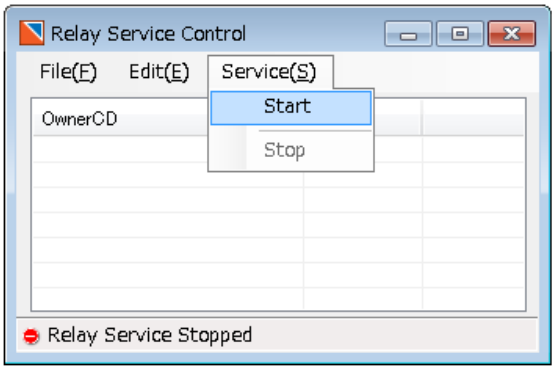
※2：カンマ区切りで複数のリターンコードを指定できます

※3：このリターンコードを返したとき、連携サービスは処理正常と見なしたうえで「前回連携した日時」を更新しません。結果は後述の代替文字列 %BeforeRelayTimeNumber%に反映されます

3.4.4 設定の反映

ユーザアカウント、対象オーナ、起動コマンドなどの設定を変更した場合、通知連携サービスを再起動して下さい。通知連携サービスはサービス起動時に設定情報を読み込んでいるため、サービスを再起動しなければ変更が反映されません。

通知連携サービスの起動停止は、ツールメニューの[Service]→[Start]または[Service]→[Stop]で実施できます。



3.5 コマンド起動方式

3.5.1 通知連携サービスの基本動作

通知連携サービスは、30 秒ごとに App Bridge センタの API を呼び出し、通知メッセージの発生を監視します。一度の API 呼び出しで取得する通知メッセージ数は 500 件であり、501 件以降の通知メッセージは次回（30 秒後）に処理対象となります。

通知連携サービスは、前回の API 呼び出しで、最後に処理した通知メッセージを前回情報として保持しており、当該通知メッセージのあとに発生した通知メッセージ以降の対象に API 呼び出しを実施します。この前回情報は通知連携サービスを停止しても保持されます。但し、前回 API 呼び出しから 240 分以上が経過していた場合、前回情報は無視され、過去 240 分間以内に発生した通知メッセージが対象となります。

3.5.2 Relay Type：All Messages

（１）コマンド起動動作

起動コマンドの設定で Relay Type に「AllMessages」を指定した場合、API で取得した通知メッセージごとにコマンドを起動します。例えば、50 件の通知メッセージが取得された場合、コマンドは 50 回連続して起動されます。

（２）代替文字列

Relay Type に「All Messages」を指定した場合、引数に下記に加え、後述の「連携データ詳細」に記述される代替文字列が使用できます。引数の代替文字列は英大文字、小文字を区別しません。

代替文字列	内容
%BeforeRelayTimeNumber%	前回連携した日時（※1）（※2）
%CurrentRelayTimeNumber%	現在日時（※2）
%RelayIntervalMilliseconds%	前回連携からの経過ミリ秒 （%CurrentRelayTimeNumber%と%BeforeRelayTimeNumber%の差）

※1：AcceptReturnCodes のリターンコードを受け取った日時（一度も受け取っていないときゼロ）
※2：1970/1/1 0:0:0.000 からの経過時間（ミリ秒）

3.5.3 Relay Type：First Message

（１）コマンド起動動作

起動コマンドの設定で Relay Type に「First Message」を指定した場合、API で取得した通知メッセージのうち、最初の１件を使用してコマンドを起動します。例えば、50 件の通知メッセージが取得された場合、コマンドは１回のみ起動されます。

（２）代替文字列

Relay Type に「First Message」を指定した場合、引数には「All Messages」と同じ代替文字列が指定できます。但し、その内容は取得した通知メッセージの最初の１件となります。

3.5.4 Relay Type：Batch

（１）コマンド起動動作

起動コマンドの設定で Relay Type に「Batch」を指定した場合、API で取得した通知メッセージで後述の「通知連携ファイル」を作成し、コマンドを起動します。例えば、50 件の通知メッセージが取得された場合、50 件の内容を含む「通知連携ファイル」が作成され、コマンドは１回のみ起動されます。コマンドの実行が終了した時点で「通知連携ファイル」は削除されます。コマンド実行終了後も当該ファイルが必要な場合、コマンドで「通知連携ファイル」を退避して下さい。

（２）代替文字列

Relay Type に「Batch」を指定した場合、引数に以下の代替文字列が指定できます。代替文字列は英大文字、小文字を区別しません。

代替文字列	内容
%RelayFile%	通知連携ファイルのパス（必要に応じ引用符で囲って下さい）
%BeforeRelayTimeNumber%	前回連携した日時（※1）（※2）
%CurrentRelayTimeNumber%	現在日時（※2）
%RelayIntervalMilliseconds%	前回連携からの経過ミリ秒 （%CurrentRelayTimeNumber%と%BeforeRelayTimeNumber%の差）

※1：AcceptReturnCodes のリターンコードを受け取った日時（一度も受け取っていないときゼロ）
※2：1970/1/1 0:0:0.000 からの経過時間（ミリ秒）

3.5.5 通知連携ファイル

（１）データ型

通知連携ファイルで使用する型を以下に示します。

型の種類	JSON 上の型	含まれるデータ
文字	string	string
数値	number	int（0～2,147,483,647）
列挙	number	int（0～2,147,483,647）
日時	number	long（1970/1/1 00:00:00.000（UTC）を起点とした経過ミリ秒）
日付	string	string（YYYYMMDD 形式のローカルタイム）
真偽	boolean	string（true または false）

（２）ファイル形式

通知連携ファイルは、以下の Json 形式を持つテキストファイルです。内容の詳細は後述の「連携データ詳細」をご参照下さい。

```
{
    “OwnerCD”: “string”,
    “RelayMessages”: [
        {
            “CreateTime”: number,
            “CreateSequence”: number,
            “NodeID”: “string”,
            “NodeCD”: “string”,
            “InstanceID”: “string”,
            “MessageType”: number,
            “OccurrenceReason”: number,
            “MessageStatus”: number,
            “MonitorType”: number,
            “OccurredTime”: number,
            “MessageID”: “string”,
            “MessageCD”: “string”,
            “Subject”: “string”,
            “Body”: “string”
        }
    ]
}
```

3.5.6 連携データ詳細

（1）データ内容

引数、通知連携ファイルに設定されるデータの内容は以下のとおりです。引数の代替文字列は英大文字、小文字を区別しません。

引数の代替文字列	通知連携ファイル	内容	備考
%OwnerCD%	OwnerCD	オーナコード	
%CreateTime%	CreateTime	メッセージが生成された日時	
%CreateSequence%	CreateSequence	同一 CreateTime 内の順序番号	
%NodeID%	NodeID	ノード ID（オーナレベル通知のとき Empty）	
%NodeCD%	NodeCD	ノードコード（詳細後述）	
%InstanceID%	InstanceID	サーバインスタンス ID（オーナレベル、ノードレベル通知のとき Empty）	
%MessageType%	MessageType	メッセージの種類 100：Information 200：Warning 900：Error	
%OccurrenceReason%	OccurrenceReason	発生理由 100：MessageAccept 200：ErrorOccurred 210：ErrorRecovery 300：WarningOccurred 310：WarningRecovery 900：SystemMessage	
%MessageStatus%	MessageStatus	メッセージの状態 100（ConfirmError）：確認済みのエラーメッセージ 200（UnconfirmError）：未確認のエラーメッセージ 300（ExceptedMessage）：抑止されたメッセージ 400（ExceptedConfirm）：確認管理されないメッセージ（警告、情報）	
%MonitorType%	MonitorType	元となった監視 1000（CPUBusy） 1100（DiskBusy） 1110（DiskSpace） 1200（PhysicalMemory） 1210（VirtualMemory） 1300（Service） 1400（Process） 2000（EndPoint_Ping） 2100（EndPoint_HTTP） 2200（EndPoint_TCPPort） 3000（API_AmazonEC2） 3100（API_AzureIaaS） 3200（API_AzurePaaS） 5000（DBSpace） 5010（DBQuery）	

		8000（EventLog） 8100（TextLog） 8900（Message） 9010（Custom）	
%OccurredTime%	OccurredTime	元となった監視データの発生日時	
%MessageID%	MessageID	メッセージを一意に示す ID（GUID）	32 文字
%MessageCD%	MessageCD	メッセージ種類を示す通知コード（後述）	10 文字
%Subject%	Subject	件名	UTF-8 換算 800 バイト以内
%Body%	Body	本文	UTF-8 換算 4000 バイト以内
%Body_D0%	-	本文をダブルクォーテーションで囲み、本文中のダブルクォーテーションを削除した値	
%Body_D2%	-	本文をダブルクォーテーションで囲み、本文中のダブルクォーテーションを反復（※1）した値	
%Body_S0%	-	本文をシンブルクォーテーションで囲み、本文中のシングルクォーテーションを削除した値	
%Body_S2%	-	本文をシンブルクォーテーションで囲み、本文中のシングルクォーテーションを反復（※1）した値	
%Body_PS%	-	本文をダブルクォーテーションで囲み、本文中の下記文字をバッククォート（`）でエスケープした値 ・$（ドルマーク） ・カンマ ・&（アンパサンド） ・スペース ・バッククォート ・ダブルクォーテーション ・シングルクォーテーション ・キャリッジリターン ・改行 ・水平タブ ・Null	PowerShell 向け変換規則

※1：1 つの文字を連続した 2 文字に変換（″→″″、'→''）

（２）ノードコード

通知連携サービスは、コマンドを起動する際、必要に応じ、API を使用し、当該オーナのノード情報を取得します。通知連携サービスは、ノード情報取得の API 使用回数を低減するため、取得したノード情報を 60 分間メモリ上にキャッシュします。このため、引数や通知連携ファイルに設定されるノードコードは最新ではない可能性があります。

（３）通知コード、件名、本文

通知コード、及び対応する件名、本文については、「App Bridge 通知メッセージガイド」をご参照下さい。

3.6 コマンド実行結果

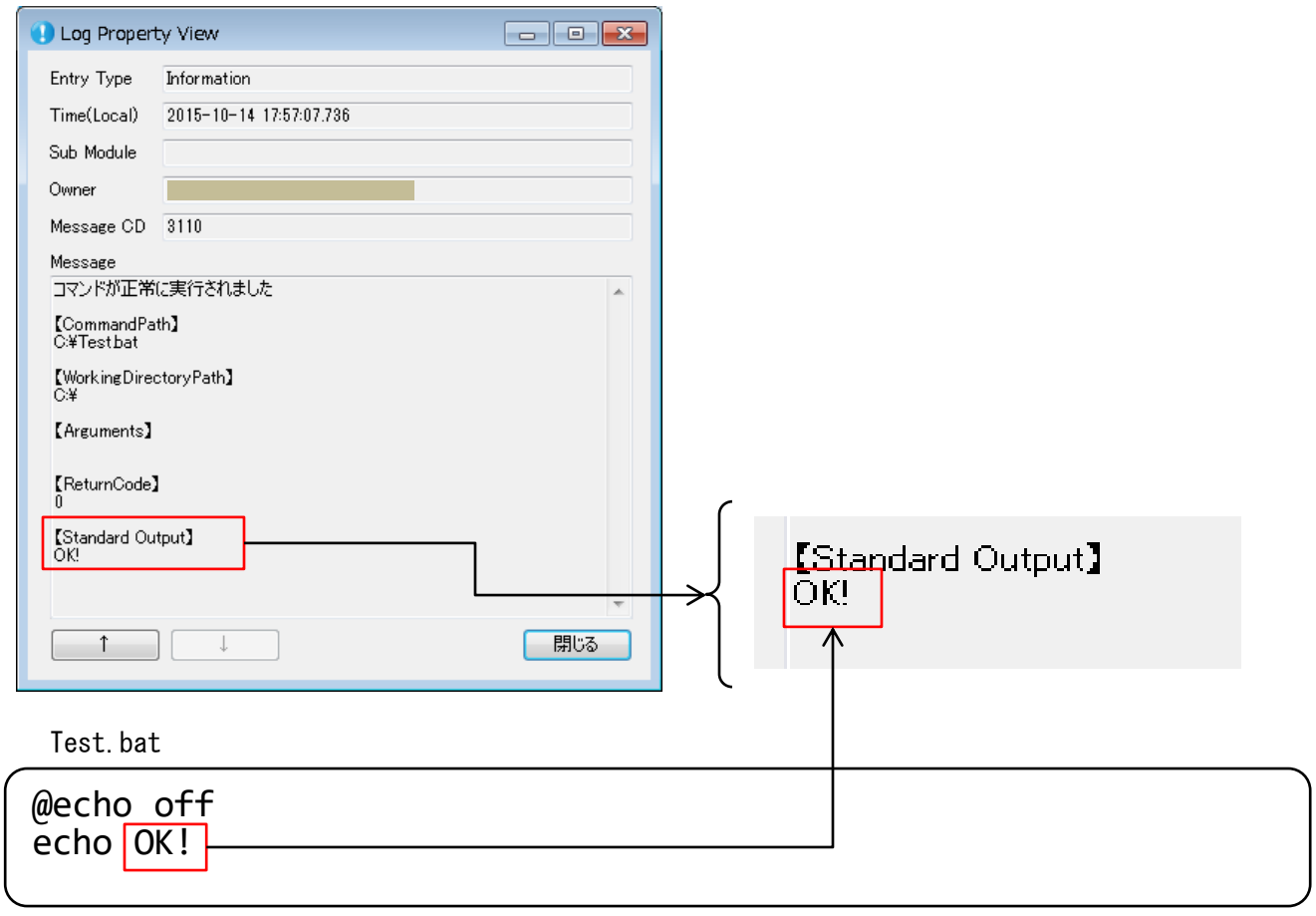
3.6.1 実行ログの確認

コマンドの実行結果は、通知連携サービスのログに出力されます。通知連携サービスのログを確認するには、メニューから、[スタート]→[App Bridge Support Tools]→[Log Viewer]を選択し、ログツールを起動します。ログツールが起動すると日付（UTC 単位）別のログファイルが表示されるため、該当のファイルを選択して下さい。

ログツールには、連携サービスの各種メッセージが記録されます。コマンドの実行に関するものは以下のとおりです。

メッセージコード	状態	取り扱い
3110	Accept Return Codes に該当するリターンコードが返された	正常
3120	Not Relay Return Codes に該当するリターンコードが返された	正常（連携なし）
8200	Accept Return Codes、Not Relay Return Codes の何れにも該当しない	異常
8210	Check Standard Error Output がチェックされており、標準エラー出力が出力された	異常

上記、4 つのメッセージでは、リターンコードに加え、標準出力、標準エラー出力が記録される。コマンドからメッセージを標準出力に出力する（ex：バッチファイルの Echo コマンド）ことで、任意の内容をログに記録することができる。



3.6.2 コマンド異常時処理

コマンド実行時にコマンドの異常終了を検知した場合、当該オーナの連携処理は一時的に中断されます。中断された連携処理は、サービスを再起動することで、再度処理対象となります。

3.6.3 障害発生時のイベントログ出力

通知連携処理で障害が発生した場合、イベントログにメッセージが出力されます。イベントログを監視することで、障害発生を検知することができます。

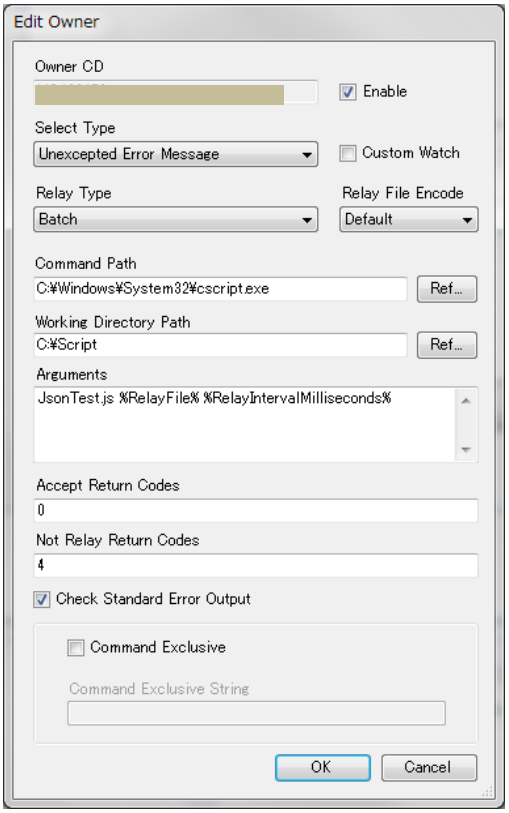
項目	内容
ログの名前	アプリケーション
ソース	UCSRelaySRV
レベル	エラー

４．Windows Script Host による連携処理

スクリプトをコマンドとして使用することで、詳細の条件分岐や多様な連携処理を実現できます。本章では、Windows Script Host（以下 WSH）を使った連携処理のサンプルを紹介します。

4.1 コマンド設定

WSH を使用する場合のコマンド設定を以下に示します。



項目	設定内容	備考
Relay Type	連携タイプを指定します	指定するコマンドに合わせて指定します（※1）
Relay File Encode	Default を指定します	Relay Type「Batch」のみ設定可能です
Command Path	コンソール向けスクリプトとして起動するため cscript.exe を指定します	通常 Windows の System32 フォルダに存在します
Working Directory Path	スクリプトファイルの存在するフォルダパスを指定します	
Arguments	最初の引数にスクリプトファイル名、2 つ目以降はスクリプトに与える引数を指定します	指定するコマンドに合わせて指定します
Accept Return Codes	正常を示すリターンコードにはゼロを指定します	
Not Relay Return Codes	正常（連携処理なし）を示すリターンコードには 4 を指定します	
Check Standard Error Output	チェックします	※2

※1：サンプルごとに値を示します
※2：WSH では実行エラーが発生した場合、リターンコードがゼロとなります。本設定をチェックすることで、実行エラーを検出できます

4.2 アプリケーションと連携する

アプリケーションと連携する場合、連携用のプログラムを呼び出します。このサンプルでは、日立製作所の運用ツール JP1/Integrated Management（以下 JP1/IM）に通知メッセージを送信する例を示します。

（１）スクリプト：JP1Run.js

```
// ---------------------------------------------
//   定数
// ---------------------------------------------
var conJP1_BRI_SRV = "MyHost";                    // ホスト名
var conJP1EVENT = "¥"C:¥¥...¥¥jevsendd.exe¥"";   // JP1 イベント送信コマンド

// ---------------------------------------------
//   パラメータ取得  1:オーナ CD 2:ノード CD 3:本文
// ---------------------------------------------
var objParm = WScript.Arguments;
if (objParm.Length == 3)
{
}
else
{
    WScript.Echo("Paramter Count Error:" + objParm.Count);
    WScript.Quit(12);
}

// ---------------------------------------------
//   JP1 イベントコマンド作成
// ---------------------------------------------
var strCommand = conJP1EVENT + " -m ¥"" + objParm(2) + "¥"" +
                 " -e SEVERITY=¥"Error¥"" +
                 " -e USER_NAME=" + objParm(0) +
                 " -e OBJECT_NAME=" + objParm(1) +
                 " -d "+ conJP1_BRI_SRV;

// ---------------------------------------------
//   コマンド実行
// ---------------------------------------------
var objWShell = WScript.CreateObject("WScript.Shell");
var runReturn = objWShell.Run(strCommand, 0, true);
if (runReturn == 0)
{
}
else
{
   WScript.Echo("Command Return Code Error:" + runReturn);
   WScript.Quit(12);
}
WScript.Quit(0);
```

（２）コマンド設定

<table>
<tr><th>項目</th><th>設定内容</th><th>備考</th></tr>
<tr><td>Relay Type</td><td>本例では、データ発生時に１度だけ連携させるため、「First」を指定します</td><td></td></tr>
<tr><td>Arguments</td><td>以下を指定します
<table>
<tr><th>引数順</th><th>引数</th><th>内容</th></tr>
<tr><td>1</td><td>JP1Run.js</td><td>スクリプト名</td></tr>
<tr><td>2</td><td>%OwnerCD%</td><td>オーナコード</td></tr>
<tr><td>3</td><td>%NodeCD%</td><td>ノードコード</td></tr>
<tr><td>4</td><td>%Body_D2%</td><td>本文（ダブルクォーテーションを繰り返した値）</td></tr>
</table></td><td>※1</td></tr>
</table>

※1：″JP1Run.js″ %OwnerCD% %NodeCD% ″%Body_D2%″

4.3 一定間隔で連携する

通知メッセージが連続して発生しても、一定時間は連携したくない場合があります。このサンプルでは、前回連携から 15 分間は連携しない例を示します。

（１）スクリプト：IntervalRun.js

```
if (objParm(0) < 15 * 60 * 1000)  //15 分間は連携しない
{
    WScript.Echo("NotRelay Interval Error");
    WScript.Quit(4);    //連携しなかった場合、リターンコード 4 を返す
}

//連携処理をここに記述します

WScript.Quit(0);     //正常に連携した場合、リターンコードゼロを返す
```

（２）コマンド設定

<table>
<tr><th>項目</th><th>設定内容</th><th>備考</th></tr>
<tr><td>Relay Type</td><td>本例では、データ発生時に 1 度だけ連携させるため、「First」を指定します</td><td></td></tr>
<tr><td>Arguments</td><td>以下を指定します
<table>
<tr><th>引数順</th><th>引数</th><th>内容</th></tr>
<tr><td>1</td><td>IntervalRun.js</td><td>スクリプト名</td></tr>
<tr><td>2</td><td>%RelayIntervalMilliseconds%</td><td>前回連携からの経過ミリ秒</td></tr>
</table></td><td></td></tr>
</table>

4.4 通知メッセージを審査する

通知メッセージの内容を審査し、連携先を変更したり、特定の条件成立時に連携することが可能です。このサンプルでは、ノードコードが「MYSERVER」の通知メッセージがあれば連携する例を示します。

（１）スクリプト：JsonRun.js

```
if (this.JSON)
{
}
else
{
    //Json がロードされていなければロードする
    // Load Libraries from https://github.com/douglascrockford/JSON-js
    eval(includeFile("JSON-js-master¥¥json2.js"));
}

var jsonrelay = JSON.parse(includeFile(objParm(0)));    //通知連携ファイルを読み込み、Json として解析
if (CheckTarget(jsonrelay))
{
}
else
{
    WScript.Echo("Data Not Found");    //ログに記録するためのエコー出力
    WScript.Quit(4);    //連携しなかった場合、リターンコード 4 を返す
}

//連携処理をここに記述します

WScript.Quit(0);    //正常に連携した場合、リターンコードゼロを返す

function includeFile(filepath)        //指定されたテキストファイルを読み込む
{
    var fso = WScript.CreateObject("Scripting.FileSystemObject");
    var fileStream = fso.openTextFile(filepath);
    var fileData = fileStream.readAll();
    fileStream.Close();
    fileStream = null;
    fso = null;
    return fileData;
}

function CheckTarget(JsonObject)
{
    for (var i1 = 0; i1 < JsonObject.RelayMessages.length; i1++)
    {
        if (CheckMessage(JsonObject.RelayMessages[i1]))
        {
            return true;
        }
    }
    return false;    //対象データが 1 つもなかった
}

function CheckMessage(JsonRelayMessage)  //このメソッドを変更すれば様々な条件に対応できる
{
    if (JsonRelayMessage.NodeCD == "MYSERVER")    //ノードコードが MYSERVER
    {
    }
    else
    {
        return false;
    }
    return true;
}
```

複数の通知メッセージの中から対象のメッセージを探して処理するためには、通知連携ファイルの読み込みが必要です。通知連携ファイルは Json 形式のため、Json ライブラリを読み込んでいます。

（２）コマンド設定

<table>
<tr><th>項目</th><th>設定内容</th><th>備考</th></tr>
<tr><td>Relay Type</td><td>本例では、通知連携ファイルを使用するため、「Batch」を指定します</td><td></td></tr>
<tr><td>Relay File Encode</td><td>Default を指定します（※1）</td><td></td></tr>
<tr><td>Arguments</td><td>以下を指定します
<table>
<tr><th>引数順</th><th>引数</th><th>内容</th></tr>
<tr><td>1</td><td>JsonRun.js</td><td>スクリプト名</td></tr>
<tr><td>2</td><td>%RelayFile%</td><td>通知連携ファイルのファイルパス</td></tr>
</table></td><td></td></tr>
</table>

※1：WSH でテキストファイルを読み込むため、OS のデフォルト文字コードを指定します。OS が日本語版の場合、シフト JIS となります。シフト JIS で表現できない文字は文字化けするためご注意下さい

4.5 サンプルのダウンロード

これらのサンプルスクリプトは、以下の URL からダウンロードすることができます。

<http://www.app-bridge.com/download/documents/SampleScript.zip>

ダウンロードしたスクリプトはサンプルであり、動作を保証するものではありません。サンプルのご使用は利用者の責任で実施下さい。

Windows Script Host の詳細については、マイクロソフト社の公開する各種ドキュメントをご参照下さい。また、連携先となるアプリケーションについては、各社のドキュメントをご参照して下さい。