お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

HEW サーバ説明添付資料

HS6400EWIW3SJ-TLB030627

1.1 HewTargetServer の概要

HEW とタイプライブラリの関係を図 1.1 に示します。

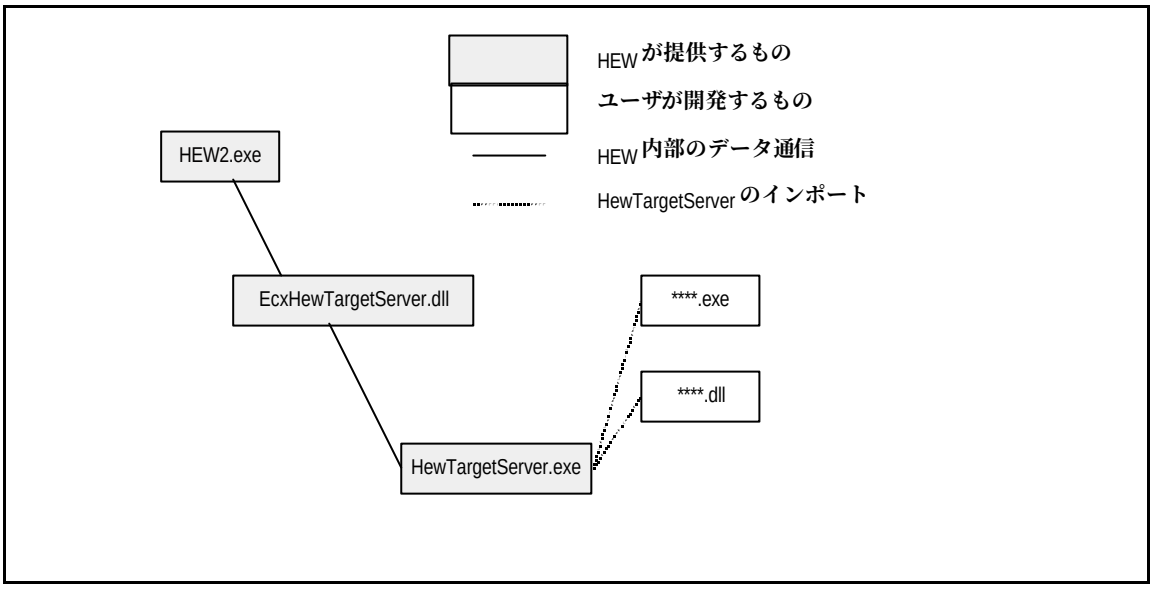

図 1.1: HEW と HewTargetServer の関係

ユーザの作成したアプリケーション(実行形式ファイル(.exe)やダイナミックリンクライブラリ(.dll))は、HewTargetServer をインポートし、インタフェース関数を呼び出すことにより、データの設定、取得が可能になります。

1.2 登録方法

HEW システムのインストールでは、HewTargetServer.exe(HEW サーバ)および EcxHewTargetServer.dll(HEW2.exe と HewTargetServer.exe 間で情報のやりとりをコントロールする DLL)を登録しません。HewTargetServer を使用するにはこれらを登録する必要があります。

(1) HewTargetServer.exe のレジストリへの登録

HEW システムのインストールディレクトリにある REGISTERSERVER.bat を実行してください。本バッチファイルを実行することにより、HewTargetServer.exe がレジストリに登録されます。

(2) EcxHewTargetServer.dll の HEW システムへの登録

(a) HEW を起動

(b) “ようこそ！”ダイアログボックスの“アドミニストレーション...”ボタンを押下

(c) “ツールアドミニストレーション”ダイアログボックスの”登録...”ボタンを押下

(d) “HEW 登録ファイルの選択”ダイアログボックスで”(HEW のインストールディレクトリ）￥System￥Sec￥HewTargetServer￥EcxHewTargetServer.hrf” または”(HEW のインストールディレクトリ)￥System￥Sec￥Hitachi HewTargetServer￥EcxHewTargetServer.hrf”を選択

以上(a)-(d)の操作により EcxHewTargetServer.dll が HEW システムに登録されます。

登録後に HewTargetServer を使用する必要がなくなった場合、登録を削除する必要があります。

(3) レジストリから HewTargetServer.exe を削除

HEW システムのインストールディレクトリにある UNREGISTERSERVER.bat を実行してください。本バッチファイルを実行することにより、HewTargetServer.exe がレジストリから削除されます。

1.3 使用方法

下記の説明は、クライアント側でどのように本タイプライブラリを使用するかのガイドです。

(1) HewTargetServer.exe のインポートをしてください。

```
#import "HewTargetServer.exe"                no_namespace
```

ここで指定する HewTargetServer.exe のパスは、HEW のインストールしたディレクトリとクライアントのディレクトリによって異なります。

(2) スマートポインタのメンバ変数として宣言してください?

```
IHewServer1Ptr    pHewServer1;
```

(3) コンストラクタによるスマートポインタの生成と初期化をしてください。

```
IHewServer1Ptr ptr(_uuidof(HewServer1));
pHewServer1 = ptr;
```

(4) デストラクタによるスマートポインタの破棄をしてください?

```
pHewServer1 = NULL;
```

(5) スマートポインタを複数使用することができます? また? すべてのポインタ値は同じです? クライアントプログラム終了時には? すべてのスマートポインタの破棄をしてください?

(6) HEW,クライアントのどちらのプログラムが先に起動されても、機能面で影響はありません。

(7) 複数のクライアントと単一の HEW が実行されているときにも、ひとつのデータを共有します。

(8) HewTargetServer.exe は、単一の HEW が動作していることを前提として設計しています。

(9) タイプライブラリとして公開しているインタフェースの呼び出し方法は? 以下のようになります?

```
try
{
        hr = pHewServer1->FUNCTION();//FUNCTION()は、HEW で公開しているインタフェースです。
}
catch(_com_error e1)
{
        if(e1.Description().length()>0)                          //カスタムエラーリターン時
                AfxMessageBox(e1.Description());                 //カスタムエラーの表示
        else
        {
                BSTR            bstrErrStr;
                try
                {                                                //HEW エラーリターン時
                        hrErr = pHewServer1->GetErrorString(e1.Error(), &bstrErrStr);
                }
                catch(_com_error  e)
                {
                }
                if(SUCCEEDED(hrErr))
                {
                        s1.Format("%s", CString(bstrErrStr));    //HEW エラーの表示
                        AfxMessageBox(s1);
                }
```

```
            else
                AfxMessageBox(e1.ErrorMessage());                    //システムエラーの表示
        }
    }
```

注意:

(a) **HEW で公開している関数を呼び出すときには、必ず try{},catch{}を使用しなくてはなりません。インタフェース関数呼び出しでエラーが発生したときには、catch{}で COM システムからの COM エラーを取得することができます。catch{}を使用しない場合に、COM エラーが発行されると、クライアントプログラムは、アプリケーションエラーを引き起こします。**

(b) **COM エラー発生したときには、以下の 3 種類のエラーがあります。**

カスタムエラー (HewTargetServer.exe が発行するエラー)

本エラーには、HEW が起動していないとき、ターゲットが接続していないとき、ロードモジュールがダウンロードされていないときに HewTargetServer.exe が呼ばれたときに返却するエラーが含まれます。

HEW エラー (HEW2.exe が発行するエラー)

HEW がエラーを返却する場合は、呼び出されたインタフェースのパラメタに不正の可能性があります。
HEW がエラーを返却したときには、GetErrorString()を呼び出すことにより、エラー内容を取得することができます。

システムエラー (COM システムが発行するエラー)

COM システムによってエラーが返却される場合は、RPC(Remote Procedure Call)環境に問題があるか、クライアントと HewTargetServer.exe のコミュニケーションに問題があります。

(c) **HEW が起動していないときに、インタフェース関数を呼び出すと以下のエラーが返却されます。**

"Hew Target is not linked up"

注意: **GetHewStatus()だけがエラーを返却せず、現在の HEW の状態を返却します。**

(d) **HewTargetServer を使用するには、OLE の初期化が必要です。**

プログラムの最初に、AfxOleInit()**を実行してください。**

(e) **HewTargetServer を使用中に、処理に長時間を要するとき [サーバー使用中] ダイアログ ボックスが表示されます。**

表示させたくない場合は、以下のサイトをご参照ください。

[VC] COM を使用した処理に長時間を要するとき [サーバー使用中] ダイアログ ボックスが表示されないようにする方法
http://support.microsoft.com/default.aspx?scid=kb;ja;248019

1.4　インタフェース

(1)　実行制御

機能	名前
プログラム を実行	GoTargetExec()
プログラム を停止	StopTargetExec()
プログラム をリセット	ResetTargetExec()
ターゲットを初期化	InitializeTarget()
プログラム をステップ実行	Step()
ステップ実行速度を設定	StepRate()
プログラム をステップオーバー	StepOver()
プログラム をステップアウト	StepOut()
プログラム が実行中かチェック	IsRunning()

(2)　レジスタ操作

機能	名前
プログラムカウンタの値を取得	GetPC()
プログラムカウンタの値をアドレスで設定	SetPCAddress()
プログラムカウンタの値を行番号で設定	SetPCSource()
PC 設定のチェック	TestSetPC()

(3)　メモリ操作

機能	名前
メモリの値を取得	GetMemory()
メモリに値を設定	SetMemory()
ダイレクトメモリ取得	GetDirectMemory()

(4)　ソフトウェアブレーク

機能	名前
ブレークポイントを設定	SetPCBreakPt()
ブレークポイントの有効/無効設定	EnableBreakPt()
ブレークポイントを削除	DeleteBreakPt()

(5)　変数ブレーク

機能	名前
データブレークポイントを設定	SetDataBreakpoint()
データブレークポイントの有効/無効設定	EnableDataBreakpoint()
データブレークポイントを削除	DeleteDataBreakpoint()

(6)　変数トレース

機能	名前
変数トレース条件を設定	SetSymbolTrace()
変数トレースの有効/無効設定	ExecuteSymbolTrace()
変数トレース条件を削除	DeleteSymbolTrace()
変数トレースの結果を指定したファイルに保存	SaveSymbolTraceData()

(7)　割り込み

機能	名前
割り込み条件を設定	SendTrigger()

(8)　シンボル操作

機能	名前
リアルタイムウォッチ	GetRealTimeWatch()
ラベル/シンボルの値を取得	GetQuickWatch()
シンボル→アドレス変換	SymbolToAddress()
アドレス→シンボル変換	AddressToSymbol()
アドレス→行番号変換	GetLineFromAddr()
行番号→アドレス変換	GetAddrFromLine()

(9)　ダウンロード

機能	名前
ターゲットプログラム をダウンロード	Download()
ターゲットプログラム をアンロード	Unload()

(10)　起動/終了

機能	名前
HEW アプリケーション を起動	InvokeHew()
HEW アプリケーション を終了	QuitHew()

(11)　ワークスペース

機能	名前
ワークスペースファイル を開く	OpenWorkspace()
ワークスペースファイル を閉じる	CloseWorkspace()
ワークスペースファイル を保存	SaveWorkspace()

(12)　構成とセッション

機能	名前
セッションファイルを保存	SaveSession()
ビルド構成を設定	SetCurrentConfiguration()
登録されているビルド構成を取得	GetConfigurations()
デバッグセッションを設定	SetCurrentSession()
登録されているデバッグセッションを取得	GetSessions()

(13)　プロジェクト

機能	名前
プロジェクトにファイルを追加	AddFile()
プロジェクトに複数のファイルを追加	AddFiles()
プロジェクトからファイルを削除	DeleteFile()
プロジェクトから複数のファイルを削除	DeleteFiles()
アクティブプロジェクトを設定	SetCurrentProject()

(14)　ビルド関係

機能	名前
プロジェクトをビルド	BuildProject()
プロジェクトをリビルド	RebuildProject()
すべての依存関係を更新	UpDateAllDependency()

(15)　その他

機能	名前
メソッド呼び出しで発生したエラー文字列を取得	GetErrorString()
ステータス取得	GetHewStatus()
ターゲット名取得	GetTargetName()

1.4.1 プログラムを実行

IHewServer1::GoTargetExec()

(1) 仕様

プログラムの実行

(2) パラメタ

なし

1.4.2 プログラムを停止

IHewServer1::StopTargetExec()

(1) 仕様

プログラムの停止

(2) パラメタ

なし

1.4.3 プログラムをリセット

IHewServer1::ResetTargetExec()

(1) 仕様

ターゲットのリセット

(2) パラメタ

なし

1.4.4 ターゲットを初期化

IHewServer1::InitializeTarget()

(1) 仕様

ターゲットの初期化

(2) パラメタ

なし

1.4.5 プログラムをステップ実行

IHewServer1::Step(

[in] DWORD _eMode,

[in] DWORD _dwStep)

(1) 仕様

ステップ実行する。

(2) パラメタ

[in] _eMode: **ステップの種類(アセンブラの命令/ソースコードの行)**

0x00000001 – **アセンブラの命令**

0x00000002 – **ソースコードの行**

[in] _dwStep: **ステップ数**

1.4.6 ステップ実行速度を設定

IHewServer1::StepRate(

[in] DWORD _dwRate)

(1) 仕様

ステップ実行速度を設定する。

(2) パラメタ

[in] _dwRate: **ステップレート**(0~6)

1.4.7 プログラムをステップオーバー

IHewServer1::StepOver(

[in] DWORD _eMode,

[in] DWORD _ dwStep)

(1) 仕様

ステップオーバー実行する。

(2) パラメタ

[in] _ eMode: **ステップの種類(アセンブラの命令/ソースコードの行)**

0x00000001 – **アセンブラの命令**

0x00000002 – **ソースコードの行**

[in] _dwStep: **ステップ数**

1.4.8 プログラムをステップアウト

IHewServer1::StepOut(

[in] DWORD _eMode)

(1) 仕様

ステップアウト実行する。

(2) パラメタ

[in] _ eMode: **ステップの種類(アセンブラの命令/ソースコードの行)**

0x00000001 – **アセンブラの命令**

0x00000002 – **ソースコードの行**

1.4.9 プログラムが実行中かチェック

IHewServer1::IsRunning(

[out] long* p_bRunning)

(1) 仕様

ユーザプログラム実行中か否かの判定

(2) パラメタ

[out]p_bRunning:**ユーザプログラム実行中のときは** 1**、それ以外は** 0

1.4.10 プログラムカウンタの値を取得

IHewServer1::GetPC(

[out] DWORD *p_dwPC)

(1) 仕様

PC 値を取得する。

(2) パラメタ

[out] p_dwPC: PC **値**

1.4.11 プログラムカウンタの値をアドレスで設定

IHewServer1::SetPCAddress(

[in] ADDR aPCAddr)

(1) 仕様

PC **値を設定**

(2) パラメタ

[in] aPCAddr:**設定する** PC **値**

1.4.12 プログラムカウンタの値を行番号で設定

IHewServer1::SetPCSource(

[in] BSTR bstrFileName,

[in] LINENO LineNum)

(1) 仕様

ファイル、行に PC **値を設定**

(2) パラメタ

[in] bstrFileName:**ファイル名**

[in] LineNum:**行**

1.4.13 PC 設定のチェック

IHewServer1::TestSetPC(

[out] long* p_bSetPCState)

(1) 仕様

PC **値設定可能か否かの判定**

(2) パラメタ

[out] p_bSetPCState:PC **値設定可能なときは** 1**、それ以外は** 0

1.4.14　メモリの値を取得

IHewServer1::GetMemory(

[in] ADDR _aBegin,

[in] ADDR _aEnd,

[in] WORD _wDisplayWidth,

[out] SAFEARRAY(BYTE)* _ppbyBuff)

(1)　仕様

指定のある開始、終了アドレス、およびアクセスサイズによりメモリ内容を取得。HEW **内部で本指定エリアのメモリ内容のデータを保持している場合は、ターゲットメモリへアクセスせずにメモリ内容を返却します。**

(2)　パラメタ

[in] _aBegin:**メモリ内容取得の開始アドレス**
[in] _aEnd:**メモリ内容取得の終了アドレス**
[in] _wDisplayWidth:**メモリへアクセスするサイズ**(1,2,4,8 **が指定可能**)
[out] _ppbyBuff:**メモリ内容**

1.4.15　メモリに値を設定

IHewServer1::SetMemory(

[in] ADDR _aBegin,

[in] ADDR _aEnd,

[in] WORD _wDisplayWidth,

[in] SAFEARRAY(BYTE) _byBuff)

(1)　仕様

指定のある開始、終了アドレス、およびアクセスサイズによりメモリ内容を設定します。

(2)　パラメタ

[in] _aBegin:**メモリ内容取得の開始アドレス。**
[in] _aEnd:**メモリ内容取得の終了アドレス**
[in] _wDisplayWidth:**メモリへアクセスするサイズ**(1,2,4,8 **が指定可能**)
[in] _byBuff:**メモリ内容**

1.4.16　ダイレクトメモリ取得

IHewServer1::GetDirectMemory(

[in] ADDR _aBegin,

[in] ADDR _aEnd,

[in] WORD _wDisplayWidth,

[out] SAFEARRAY(BYTE)* _ppbyBuff)

(1)　仕様

指定した開始、終了アドレス、およびアクセスサイズによりメモリ内容を取得。HEW **内部で本指定エリアのメモリ内容のデータを保持している、いないにかかわらず、ターゲットメモリへアクセスしてメモリ内容を返却します。**

(2)　パラメタ

[in] _aBegin:**メモリ内容取得の開始アドレス。**
[in]_aEnd:**メモリ内容取得の終了アドレス**
[in]_wDisplayWidth:**メモリへアクセスするサイズ**(1,2,4,8 **が指定可能**)
[out]_ppbyBuff:**メモリ内容**

1.4.17　ブレークポイントを設定

IHewServer1::SetPCBreakPt(

[in] ADDR _aPCBreakAddr,

[out] BHANDLE* p_BHandle)

(1)　仕様

指定アドレスにブレークポイントを設定し、そのハンドル値を返却します。

(2)　パラメタ

[in] aPCBreakAddr:**アドレス値**
[out]p_BHandle:**ブレークポイントのハンドル値**

1.4.18　ブレークポイントの有効/無効設定

IHewServer1::EnableBreakPt(

[in] BHANDLE BHandle ,

[in] VARIANT_BOOL bEnable)

(1)　仕様

ブレークポイントのハンドル値から、ブレークポイントの有効化、無効化を設定します。

(2)　パラメタ

[in] _BHandle:**ブレークポイントのハンドル値**
[in] _bEnable:**有効化**(VARIENT_TRUE),**無効化**(VARIENT_FALSE)

1.4.19　ブレークポイントを削除

IHewServer1::DeleteBreakPt(

[in] BHANDLE BHandle)

(1)　仕様

ブレークポイントのハンドル値のブレークポイントを削除します。

(2)　パラメタ

[in] BHandle:**ブレークポイントのハンドル値**

1.4.20　データブレークポイントを設定

IHewServer1::SetDataBreakpoint (

[in] DWORD _aSymbol,

[in] DWORD _eSize,

[in] DWORD _eType,

[in] DWORD _dwData,

[out] DWORD *p_dwBreakDataNo)

(1)　仕様

データブレークポイントを設定する。

(2)　パラメタ

[in] _aSymbol: **シンボルのアドレス**
[in] _eSize: **シンボルのサイズ**(1/2/4)

0x00000001　– 1

0x00000002　– 2

0x00000004　– 4

[in] _eType: **ブレークする種類**(Equal/Not Equal)

0x00000001　– Equal

0x00000002　– Not Equal

[in] _ dwData: **シンボルの値**
[out] p_dwBreakDataNo: **変数ブレークの番号**

1.4.21　データブレークポイントの有効/無効設定

IHewServer1::EnableDataBreakpoint(

[in] DWORD dwBreakDataNo,

[in] VARIANT_BOOL _bEnable)

(1)　仕様

データブレークポイントの有効/無効を設定する。

(2)　パラメタ

[in] dwBreakDataNo: **変数ブレークの番号**
[in] _ bEnable: 有効(True)/ 無効(False)

1.4.22　データブレークポイントを削除

IHewServer1::DeleteDataBreakpoint (

[in] DWORD　dwBreakDataNo)

(1)　仕様

データブレークポイントを削除する。

(2)　パラメタ

[in] dwBreakDataNo: **変数ブレークの番号**

1.4.23　変数トレース条件を設定

IHewServer1::SetSymbolTrace (

[in] ADDR _ aSymbol,

[in] DWORD _ eCondition,

[in] DWORD _ eSize,

[in] DWORD _ eType,

[in] DWORD _ dwData,

[out] DWORD * p_ dwTraceNo)

(1)　仕様

トレース条件を設定する。

(2)　パラメタ

[in] _aSymbol: **シンボル名のアドレス**

[in] _eCondition: **トレース条件**(Read/Write)

0x00000001　– Read

0x00000002　– Write

0x00000003　– Read_Write

[in] _eSize: **シンボルのサイズ**(1/2/4)

0x00000001　– 1

0x00000002　– 2

0x00000004　– 4

[in] _eType: **トレースする種類**(Equal/Not Equal/No Specific)

0x00000001　– Equal

0x00000002　– Not Equal

0x00000003　– Not Specified

[in] _dwData: **シンボルの値**

[out] p_dwTraceNo: **変数トレースの番号**

1.4.24　変数トレースの有効/無効設定

IHewServer1::ExecuteSymbolTrace (

[in] VARIANT_BOOL _bEnable)

(1)　仕様

変数トレースを有効/無効設定する。

(2)　パラメタ

[in] _ bEnable: **有効**(True)/**無効**(False)

1.4.25　変数トレース条件を削除

IHewServer1::DeleteSymbolTrace (

[in] DWORD _ dwTraceNo)

(1)　仕様

トレース条件を削除する。

(2)　パラメタ

[in] _dwTraceNo: **削除する変数トレースの番号**

1.4.26　変数トレースの結果を指定したファイルに保存

IHewServer1::SaveSymbolTraceData (

[in] BSTR _bstrFileName)

(1)　仕様

変数トレースの結果を指定したファイルに保存する。
変数トレース結果のフォーマットは、付録 A を参照してください。

(2)　パラメタ

[in] _bstrFileName: **変数トレースデータの保存ファイル**

1.4.27　割り込み条件を設定

IHewServer1::SendTrigger (

[in] DWORD _dwTriggerNo,

[in] DWORD _dwTriggerType1,

[in] DWORD _dwTriggerType2,

[in] DWORD _dwPriority)

(1)　仕様

トリガー条件を設定する。

(2)　パラメタ

[in] _ dwTriggerNo: **トリガー番号**
[in] _ dwTriggerType1: **トリガー割り込み条件** 1
[in] _ dwTriggerType2: **トリガー割り込み条件** 2
[in] _ dwPriority : **割り込み優先順位**(0~17)

1.4.28　リアルタイムウォッチ

IHewServer1::GetRealTimeWatch (

[in] ADDR _aSymbol,

[in] DWORD _eSize,

[out] DWORD *p_dwValue)

(1)　仕様

リアルタイムウォッチする。

(2)　パラメタ

[in] _aSymbol: **シンボル名のアドレス**
[in] _eSize: **シンボルのサイズ**(1/2/4)

0x00000001　– 1

0x00000002　– 2

0x00000004　– 4

[out] p_dwValue: **シンボルの値**

1.4.29　ラベル/シンボルの値を取得

IHewServer1::GetQuickWatch(

[in] BSTR bstrVarName,

[out] DWORD* p_dwValueSize,

[out] BSTR* bstrByValue,

[out] EObjectTypeServer* p_eType,

[out] BSTR* bstrTypeName,

[out] BSTR* bstrVariableAllocation)

(1)　仕様

変数名と現在の PC 値から、変数のサイズ、変数値、型、割付領域に変換します。

(2)　パラメタ

[in] _bstrVarName:**変数名**
[out] _pdwValueSize:**変数のサイズ**
[out] bstrByValue:**変数値の文字列**
[out] p_eType:**変数型**
[out] bstrTypeName:**変数型の文字列**
[out] bstrVariableAllocation:**変数割付領域の文字列**

1.4.30　シンボル→アドレス変換

IHewServer1::SymbolToAddress(

[in] BSTR bstrSymbolName,

[out] ADDR* p_aSymbolAddr)

(1)　仕様

シンボル名から対応するアドレス値へ変換します。

(2)　パラメタ

[in] _bstrSymbolName:**シンボル名**
[out] _paSymbolAddr:**アドレス**

1.4.31　アドレス→シンボル変換

IHewServer1::AddressToSymbol(

[in] ADDR aSymbolAddr,

[out] BSTR* p_bstrSymbolName)

(1)　仕様

アドレス値から対応するシンボル名へ変換します。

(2)　パラメタ

[in] _aSymbolAddr:**アドレス**
[out]p_bstrSymbolName:**シンボル名**

1.4.32　アドレス→行番号変換

IHewServer1::GetLineFromAddr(

[in] ADDR _aLineAddr,

[out] BSTR* p_bstrFileName,

[out] LINENO* p_LineNo)

(1)　仕様

アドレス値から対応するファイル、行へ変換します。

(2)　パラメタ

[in] _aLineAddr:**アドレス**
[out]p_bstrFileName:**ファイル名**
[out]p_LineNo:**行番号**

1.4.33　行番号→アドレス変換

IHewServer1::GetAddrFromLine(

[in] BSTR bstrFileName,

[in] LINENO LineNo,

[out] ADDR* p_aLineAddr)

(1)　仕様

ファイル、行から対応するアドレス値へ変換します。

(2)　パラメタ

[in] bstrFileName:**ファイル名**
[in] LineNo:**行番号**
[out] p_aLineAddr:**アドレス**

1.4.34　ターゲットプログラムをダウンロード

IHewServer1::Download(

[in] BSTR _bstrFileName)

(1)　仕様

ロードモジュールをダウンロードする。

(2)　パラメタ

[in] _ bstrFileName:**ロードモジュール(パス名を含む)**

1.4.35　ターゲットプログラムをアンロード

IHewServer1::Unload(

[in] BSTR _bstrFileName)

(1)　仕様

ロードモジュールをアンロードする。

(2)　パラメタ

[in] _ bstrFileName:**アンロードするモジュール(パス名を含む)**

1.4.36　HEW アプリケーションを起動

IHewServer1::InvokeHew ()

(1)　仕様

HEW **プログラムを起動する。(ワークスペースは、開かない。)**

(2)　パラメタ

なし

InvokeHew()**を使う場合、環境変数の** PATH **に** HEW **のインストールディレクトリを追加してください。**

1.4.37　HEW アプリケーションを終了

IHewServer1::QuitHew()

(1)　仕様

HEW **アプリケーションを終了します。**

(2)　パラメタ

なし

1.4.38　ワークスペースファイルを開く

IHewServer1::OpenWorkspace(

[in] BSTR _bstrFileName)

(1)　仕様

ワークスペースファイルを開く。

(2)　パラメタ

[in] _bstrFileName:**ファイル名(パス名を含む)**

1.4.39　ワークスペースファイルを閉じる

IHewServer1::CloseWorkspace(
[in] DWORD _dwIgnoreChanges)

(1)　仕様

ワークスペースファイルを閉じる。

(2)　パラメタ

[in] _ dwIgnoreChanges:
0x00000000-　**変更があった場合、ワークスペースを閉じない**
0x00000001:**変更を保存せずに、ワークスペースを閉じる**

1.4.40　ワークスペースファイルを保存

IHewServer1::SaveWorkspace()

(1)　仕様

ワークスペースファイルを保存する。

(2)　パラメタ

なし

1.4.41　セッションファイルを保存

IHewServer1::SaveSession()

(1)　仕様

セッションファイルを保存する。

(2)　パラメタ

なし

1.4.42　ビルド構成を設定

IHewServer1::SetCurrentConfiguration(
[in] BSTR _bstrConfigurɑtion)

(1)　仕様

現在有効なビルド構成を設定する。

(2)　パラメタ

[in] _ bstrConfiguration: **ビルド構成名**

1.4.43　登録されているビルド構成を取得

IHewServer1::GetConfigurations(

[out] BSTR *p_bstrConfigurations)

(1)　仕様

プロジェクトのあるビルド構成をすべて取得する。

(2)　パラメタ

[out] p_bstrConfigurations: **ビルド構成名(名前が複数ある場合は、カンマで区切る)**
(例)”Debug, Release, SimDebug_SH-4”

1.4.44　デバッグセッションを設定

IHewServer1::SetCurrentSession(

[in] BSTR _bstrSession)

(1)　仕様

現在有効なデバッグセッションを設定する。

(2)　パラメタ

[in] _ bstrSession: **デバッグセッション名**

1.4.45　登録されているデバッグセッションを取得

IHewServer1::GetSessions(

[out] BSTR *p_bstrSessions)

(1)　仕様

プロジェクトにある全デバッグセッションを取得する。

(2)　パラメタ

[out] p_bstrSessions: **デバッグセッション名(名前が複数ある場合は、カンマで区切る)**
(例)” DefaultSession, SimSessionSH-4”

1.4.46　プロジェクトにファイルを追加

IHewServer1::AddFile(

[in] BSTR _bstrFileName)

(1)　仕様

現在有効なプロジェクトにファイルを追加する。

(2)　パラメタ

[in] _bstrFileName:**ファイル名**

1.4.47　プロジェクトに複数のファイルを追加

IHewServer1::AddFiles(

[in] BSTR _bstrFileName)

(1)　仕様

プロジェクトに複数のファイルを追加する。

(2)　パラメタ

[in] _bstrFileName:**ファイル名(ファイルが複数ある場合は、カンマで区切る)**

1.4.48　プロジェクトからファイルを削除

IHewServer1::DeleteFile(

[in] BSTR _bstrFileName)

(1)　仕様

現在有効なプロジェクトからファイルを削除する。

(2)　パラメタ

[in] _bstrFileName:**ファイル名**

1.4.49　プロジェクトから複数のファイルを削除

IHewServer1::DeleteFiles(

[in] BSTR _bstrFileName)

(1)　仕様

現在有効なプロジェクトから複数のファイルを削除する。

(2)　パラメタ

[in] _bstrFileName:**ファイル名(ファイルが複数ある場合は、カンマで区切る)**

1.4.50　アクティブプロジェクトを設定

IHewServer1::SetCurrentProject(

[in] BSTR _bstrProjectName)

(1)　仕様

アクティブプロジェクトに設定する。

(2)　パラメタ

[in] _bstrProjectName:**プロジェクト名**

1.4.51　プロジェクトをビルド

IHewServer1::BuildProject()

(1)　仕様

プロジェクトをビルドする。

(2)　パラメタ

なし

1.4.52　プロジェクトをリビルド

IHewServer1::RebuildProject()

(1)　仕様

プロジェクトをリビルドする。

(2)　パラメタ

なし

1.4.53　すべての依存関係を更新

IHewServer1::UpDateAllDependency()

(1)　仕様

すべての依存関係を更新する。

(2)　パラメタ

なし

1.4.54　メソッド呼び出しで発生したエラー文字列を取得

IHewServer1::GetErrorString(
[in] HRESULT _lError,
[out] BSTR* _pbstrError)

(1)　仕様

エラー番号に対応したエラーメッセージを取得します。

(2)　パラメタ

[in] _lError:**エラー番号**
[out] _pbstrError: **エラーメッセージ**

1.4.55　ステータス取得

IHewServer1::GetHewStatus(

[out] int* p_iTargetReset,

[out] int* p_iTargetExecStatus,

[out] int* p_iMemoryReset,

[out] int* p_iRegisterReset,

[out] int* p_iLInkStatus,

[out] int* p_iPlatformInitialize,

[out] int* p_iLoadingStatus)

(1)　仕様

現在の HEW **の状態を取得します。本関数を呼び出すことにより** HEW **の状態を取得できます。**

(2)　パラメタ

[out] p_iTargetReset:**ターゲットリセット時には** 1 **が返却、それ以外は** 0*
[out] p_iTargetExecStatus:**ユーザプログラム実行中は** 1 **が返却、それ以外は** 0
[out] p_iMemoryReset:**メモリ内容が更新されているときには** 1 **が返却、それ以外は** 0*
[out] p_iRegisterReset:**レジスタ値が更新されているときには** 1 **が返却、それ以外は** 0*
[out] p_LInkStatus:**ターゲット接続時には** 1 **が返却、それ以外は** 0
[out] p_iPlatformInitialize:**ターゲット初期化後には** 1 **が返却、それ以外は** 0*
[out] p_iLoadingStatus:**プログラムロード後には** 1 **が返却、それ以外は** 0
*:**本関数が呼び出されると、本フラグは** 0 **にリセットされます。**

1.4.56　ターゲット名取得

IHewServer1::GetTargetName(

[out] BSTR* p_bstrName)

(1)　仕様

現在接続しているターゲット名を取得します。

(2)　パラメタ

[out] p_bstrName:**ターゲット名**

1.5 HEW3.0 で取得可能なイベント

イベント種類	イベント発行タイミング
Event1_ToClient_TargetReset	ターゲットリセット時に発行
Event2_ToClient_Go	ターゲットプログラム実行時に発行
Event3_ToClient_Stop	ターゲットプログラム停止時に発行
Event4_ToClient_MemoryReset	メモリ更新時に発行
Event5_ToClient_RegisterReset	レジスタ更新時に発行
Event6_ToClient_LinkUp	ターゲットリンクアップ時に発行
Event7_ToClient_LinkDown	ターゲットリンクダウン時に発行
Event8_ToClient_PlatformInitialize	プラットフォーム初期化時に発行
Event9_ToClient_Download	プログラムダウンロード時に発行
Event10_ToClient_Unload	プログラムアンロード時に発行

イベントの取得方法については、付録 B を参照してください。

付録 A　トレース結果のフォーマット

トレース結果は、以下の内容をスペースで区切って出力します。
-アクセスした時間(シミュレータではサイクル数)
-アクセスしたアドレス
-アクセス属性(Read/Write/Read_Write)
-アクセス値
-アクセスサイズ

サンプル

```
1287539 0XFFFE5DC Write 0XEA 1
1287553 0XFFFE5DC Write 0X30 1
1288170 0XFFFE5DC Write 0XEA 1
1445327 0XFFFE5DC Write 0XE0 1
1445341 0XFFFE5DC Write 0X30 1
1445958 0XFFFE5DC Write 0XE0 1
1605377 0XFFFE5DC Write 0X4C 1
1605391 0XFFFE5DC Write 0X30 1
1606008 0XFFFE5DC Write 0X4C 1
1760876 0XFFFE5DC Write 0XF6 1
1760890 0XFFFE5DC Write 0X30 1
1761507 0XFFFE5DC Write 0XF6 1
1920063 0XFFFE5DC Write 0X6A 1
1920077 0XFFFE5DC Write 0X30 1
1920694 0XFFFE5DC Write 0X6A 1
2079569 0XFFFE5DC Write 0XF3 1
2079583 0XFFFE5DC Write 0X30 1
2080200 0XFFFE5DC Write 0XF3 1
```

付録 B　イベントの取得方法のサンプル

ダイアログベースのアプリケーション(Client)を作成した場合、以下の方法でイベント(Event3_ToClient_Stop)を取得できます。本サンプルでは、HEW でターゲットプログラムが停止したとき、ダイアログを表示します。

(1)　StdAfx.h

StdAfx.h に以下の行を追加

```
#import "HewTargetServer.exe"　　no_namespace named_guids
```

(2)　Client.cpp

CClientApp::InitInstance()に AfxOleInit()を追加

(3)　ClientDlg.cpp

#include "Afxctl.h"を追加

CClientDlg::CClientDlg()に以下の 7 行を追加

```
try{
    IHewServer2Ptr ptr(_uuidof(HewServer1));  //creating smart pointer from new i/f: IhewServer2
    pHewServer1 = ptr;
}
catch(_com_error e)
{
}
```

(4)　ClientDlg.h

#include "EventHandler.h"を追加

以下の 5 行を追加

```
public:
    IHewServer2Ptr    pHewServer1;                    //using smart pointer from new interface: IHewServer2
protected:
        CEventHandler*        m_pHandler;
        DWORD                 m_dwCookie;
```

(5) EventHandler.cpp

以下の内容のファイルを追加

```
// EventHandler.cpp : implementation file

#include "stdafx.h"
#include "EventHandler.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CEventHandler, CCmdTarget)

CEventHandler::CEventHandler()
{
	EnableAutomation();
}

CEventHandler::~CEventHandler()
{
}

void CEventHandler::OnFinalRelease()
{
	CCmdTarget::OnFinalRelease();
}

BEGIN_MESSAGE_MAP(CEventHandler, CCmdTarget)
	//{{AFX_MSG_MAP(CEventHandler)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CEventHandler, CCmdTarget)
	//{{AFX_DISPATCH_MAP(CEventHandler)
	DISP_FUNCTION_ID(CEventHandler, "Event3_ToClient_Stop", 3, OnHewStatusStop, VT_EMPTY, 0)
	//}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

BEGIN_INTERFACE_MAP(CEventHandler, CCmdTarget)
	INTERFACE_PART(CEventHandler, DIID__IHewServer2Events, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEventHandler message handlers
void CEventHandler::OnHewStatusStop()
{
	AfxMessageBox("Event3_ToClient_Stop");
}
```

(6) EventHandler.h

以下の内容のファイルを追加

```
#if !defined(AFX_EVENTHANDLER_H__0F96FDDD_7167_457D_8069_73D9AEFCDF49__INCLUDED_)
#define AFX_EVENTHANDLER_H__0F96FDDD_7167_457D_8069_73D9AEFCDF49__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// EventHandler.h : header file

/////////////////////////////////////////////////////////////////////////////
// CEventHandler command target

class CEventHandler : public CCmdTarget
{
        DECLARE_DYNCREATE(CEventHandler)
        CEventHandler();                // protected constructor used by dynamic creation
public:
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CEventHandler)
        public:
        virtual void OnFinalRelease();
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CEventHandler();

        // Generated message map functions
        //{{AFX_MSG(CEventHandler)
                afx_msg void OnHewStatusStop();                 //event call back function
        //}}AFX_MSG

        DECLARE_MESSAGE_MAP()
        // Generated OLE dispatch map functions
        //{{AFX_DISPATCH(CEventHandler)
                // NOTE - the ClassWizard will add and remove member functions here.
        //}}AFX_DISPATCH
        DECLARE_DISPATCH_MAP()
        DECLARE_INTERFACE_MAP()
};
/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif
// !defined(AFX_EVENTHANDLER_H__0F96FDDD_7167_457D_8069_73D9AEFCDF49__INCLUDED_)
```

以上