ビデオウォールコントローラ

DXN6000 シリーズ

<コマンドガイド>

取扱説明書 Ver.1.0.0

- この度は、本製品をお買い上げいただきまして誠にありがとうございます。
- 本製品の性能を十分に引き出してご活用いただくために、ご使用前に必ずこの取扱説明書をお読みください。また、お読みになった後は、本製品近くの見やすい場所に保管してください。

Arvanics Corporation

この取扱説明書をお読みいただく前に

- この取扱説明書の無断転載を禁じます。
- お客様がお持ちの製品のバージョンによっては、この取扱説明書に記載される外観図やメニュー項目などが、一部異なる場合がありますのでご了承ください。
- 取扱説明書は改善のため、事前の予告なく変更することがあります。最新の取扱説明書は、弊社のホームページからダウンロードすることができます。
  http://www.arvanics.com

取扱説明書の分冊構成

この取扱説明書は、目的に応じて分冊で提供しています。必要に応じて、各取扱説明書をお読みください。なお、クイックスタートガイドおよびコマンドガイドについては、弊社ホームページからのダウンロード提供のみになります。

■ユーザーズガイド
[目的]
・設置し、周辺機器と接続をする。
・入出力調整や設定などをする。

■クイックスタートガイド
[目的]
・簡単な操作方法を知る。

■コマンドガイド (本書)
[目的]
・シリアル通信および LAN 通信などによる外部制御をする。

商標について

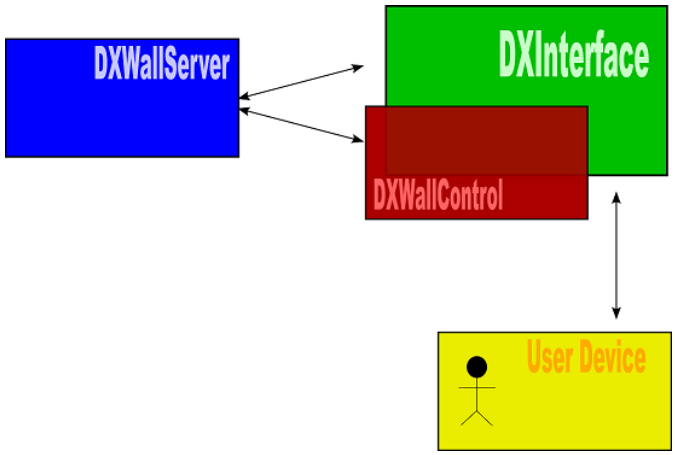
- HDMI、High-Definition Multimedia Interface、および HDMI ロゴ は、米国およびその他の国における HDMI Licensing, LLC の商標または、登録商標です。
- その他、記載されている会社名、製品名は、各社の登録商標または商標です。なお、本文中において、®マークや™マークを省略している場合があります。

目次

1. DXWALL CONTROL システムのオペレーション

1.1. DXWall ソフトウェアシステムの構造

DXWall ソフトウェアシステムは、クライアント・サーバー方式を採用しています。本機は DXWall Server と呼ばれるサーバーソフトウェアを実行します。DXWallControl のようなクライアントアプリケーションは、本機のローカル、または LAN に接続されているコンピュータ上で実行可能です。クライアントとサーバー間の通信は、TCP/IP ソケットインターフェースを介して行います。



1.1.1. DXWall Server

DXWallServer は本機で動作する Windows OS の サービスで、システムの起動時に自動的に起動します。DXWallServer は DXWall システム の核となる役割を持っています。DXWallServer は設定した TCP ポートで待ち受け、DXWallControl 等のクライアントアプリケーションからの TCP 接続を受け入れます。DXWallServer はクライアントからのリクエストを受け入れ、要求されたタスクを実行します。
クライアントの要求を受けて、DXWallServer は以下の様々なシステムデバイスをコントロールします。

- グラフィックデバイスのドライバの管理
- DXWall ディスプレイデバイスを制御し、ビデオウィンドウを管理
- DXFG フレームグラバーデバイスの制御
- 内部または外部のマトリックススイッチャへのコマンドの送信

1.1.2.DXWallControl クライアント

DXWallControl は、ローカルまたはリモートで動作するグラフィカルインターフェースです。このアプリケーションは、様々なタスクに対してそれぞれ異なったユーザーインターフェースを提供します。DXWall Server との接続を確立した後、DXWallControl から DXWallServer へとデータを送信し、その応答を受け取るという流れで通信を行います。

DXWall ソフトウェアは、以下のタスクを行うように設計されています。

管理タスク
- ユーザー権限管理
- DXWallServer へのログイン、ログアウト

映像コントロールタスク
- ビデオウィンドウの操作
- LAN を介したオペレーターパソコン上のコンテンツの表示
- Windows アプリケーションを表示するウィンドウの操作

映像シナリオタスク
- レイアウトを含んだシナリオファイルの作成と編集
- 編集されたレイアウトをファイルに保存
- 作成済みビデオレイアウトの表示
- シナリオに割り当てられたイベントやタイミングの処理

リモートオペレータータスク
- オペレーターパソコンの入力デバイスを使用し、本機のリモートコントロール
- ウォール上に配置されたウィンドウ位置やサイズの概要表示
- グラフィックウィンドウのウォールプレビュー

1.1.3.DXInterface モジュール

DXInterface プログラムは、DXWall ソフトウェアシステムの独立したサービスです。外部タッチパネルデバイスなどとのインターフェースや、レイアウト変更のスケジュール実行が可能です。

DXInterface モジュールには以下の機能があります。

スケジューリングタスク
- ひとつまたは複数のウォールレイアウトのためのシナリオファイルの実行
- 日時によるイベントの実行
- 日、週、月のタイミングによる基づいたスケジュールの管理

RS-232C シリアルと TCP/IP の制御
- 通信パラメータの設定
- シナリオファイルとイベントの設定
- 定義済み文字列に基づくイベント作成
- ウィンドウプログラミング
- AMX、Crestron、Cue タッチパネルのインターフェース

SNMP 制御

- 通信パラメータの設定
- シナリオファイルとイベントの指定
- 定義済み SNMP トラップに基づくイベント作成

HTTP インターフェース

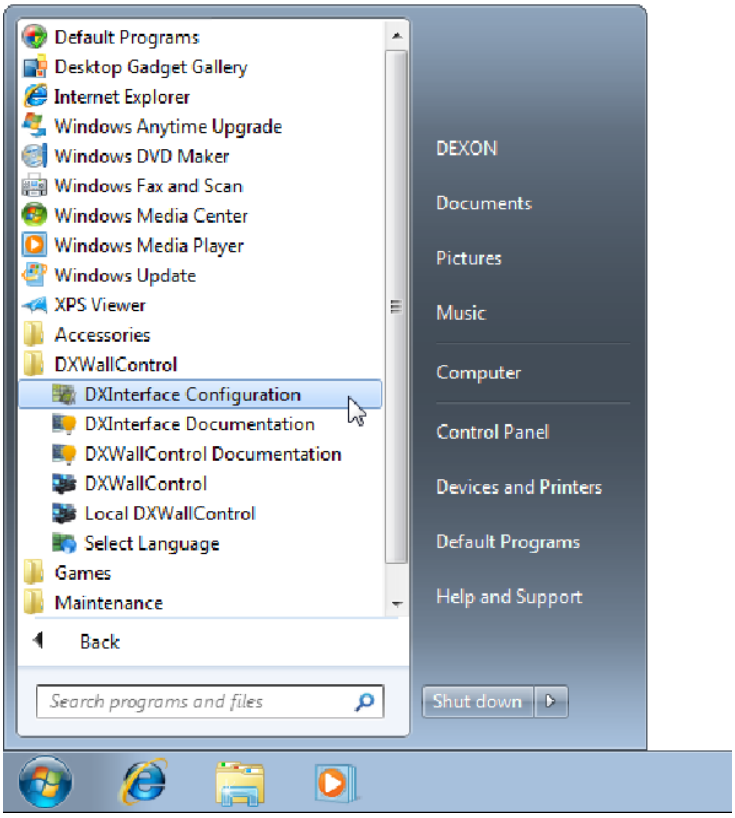
- HTTP リクエストによるイベント作成とレイアウト制御
- シナリオに基づいた動的なページの生成
- ページデザインのカスタマイズ

プログラミング

- 使いやすいユニバーサルプロトコルとスクリプト言語 (WIL - Window Interface Language)
- シリアルと TCP でのウォール操作

1.2. DXInterface の開始

DXInterface プログラムは DXWallControl ソフトウェアシステムの一部であり、DXWallControl とシナリオプログラムと連携して動作します。[スタート] → [すべてのプログラム] → [DXWallControl] → [DXInterface Configuration] から起動できます。



2. 構成と使い方

2.1. はじめに

DXWall Interface は外部デバイスやソフトウェアシステムと DXWallServer を簡単に連携させます。
インターフェースは DXWall のサーバー部とは異なる通信チャンネルを制御します。
インターフェースと DXWall システム間の全てのメッセージ・コマンドは、このチャンネルを通して送受信されます。ユーザーはアプリケーション固有のインターフェースモジュールを使用することができます。これらは様々なシステム構成において非常に高い柔軟性をもたらします。
モジュールはカスタマイズが可能でほとんどのデバイスの要求を満たすことができます。
DXWall Interface が提供するモジュールは以下になります。

- RS-232 インターフェース
- TCP/IP インターフェースとコマンドラインインターフェース
- SNMP トラップをキャプチャするインターフェース
- HTTP インターフェース：
  HTTP リクエストでウォールを制御し、ブラウザ経由でレイアウト変更が可能

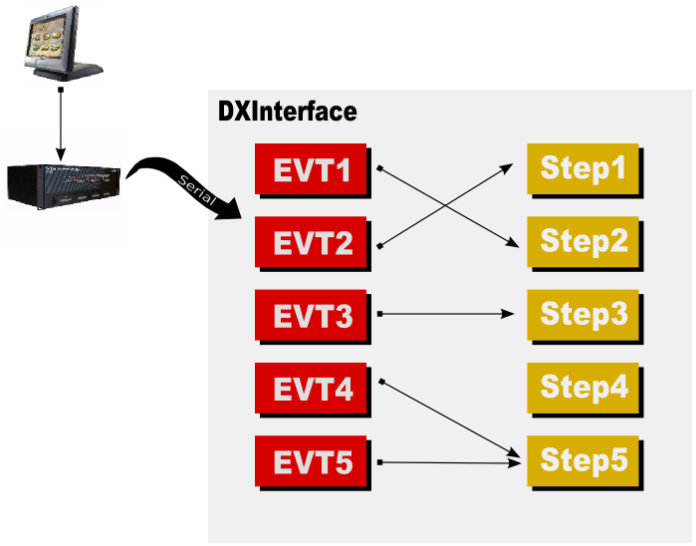
2.1.1. シンプルなビデオウォールの操作

ウォールの制御には 3 つのレベルがあり、外部デバイスから制御が可能です。
以下のプログラミング方法を使用することで、イーサネットや RS-232 で作成済みのレイアウトを呼び出すシンプルな制御が可能です。

このプログラミングでは、クライアントは**「イベント (EVT)」**コマンドを送信します。各イベントは本システムで **「ステップ (Step)」** と呼ばれる定義済みのレイアウトに関連付けられます。これらのステップは DXWallControl アプリケーションで簡単に作成、編集が可能です。
ユーザーが外部デバイスから Raw フォーマットコマンド、または EVT プロトコルコマンドを送信することで、ウォールのレイアウトを切り換えます。外部デバイスから送信されるコマンドは、DXInterface アプリケーションにて作成済みのステップと関連付けができます。作成済みのステップとは、1 つまたは複数のシナリオファイルに保存されたレイアウトのことです。
デバイスが送信するコマンドを変更することなく、レイアウトの修正やステップとの関連付けの修正を後から行うことが可能です。

- **コマンド**
  コマンド送信のトリガーには、時刻によるスケジュール実行、外部デバイスのボタン押下、センサーイベント、ウェブブラウザによる呼び出し、SNMP トラップなどが考えられます。
- **ステップ / レイアウト**
  ステップとは、シナリオファイルに格納されているレイアウトのことです。ステップは、シリアル番号 (#1、#2、#3) またはユーザー定義の外部イベントコードで識別されます。
- **シナリオファイル**
  ステップを複数格納できる単一ファイルです。自動実行を設定することも可能です。

テキストベースのプロトコルとバイナリプロトコル、そしてチャンネルの状態を取得する拡張機能を用意しています。もしより高いレベルの柔軟性を求める場合は、独自のプロトコルメッセージを定義することもできます。イベントは HTTP、SNMP、スケジューラ・モジュールによっても作成可能です。



このレベルのウォールプログラミングを使用する場合は、以下の章を参照してください。

2.1.2.ウィンドウプロパティのカスタマイズ

作成済みレイアウトを使った画面構成の変更より高度なことをしたいが、シンプルな通信コマンドでの制御は行いたい場合は、以下の方法がお勧めです。
前述したすべての機能を利用し、設定された入力チャンネルのリスト取得や、ビデオやアプリケーションのウィンドウの開閉ができます。バイナリ、またはテキストプロトコルを使用したインターフェースの拡張が可能です。ユーザーは TCP クライアントから手動でコマンドの送信が可能です。また、高速なプロセッサと多くのメモリを積んだデバイスだけでなく、低帯域のシリアル通信を使う組み込み機器も使用できます。

一意の ID、リストインデックス、または設定した入力チャンネルの名前を使用して映像を制御します。名前は UTF-8 文字列で設定できます。必要に応じて、開いているビデオウィンドウにキーワードを付けることで、消費メモリの節約が可能です。プロトコルコマンドを使って一般的なプロパティである、ポジション、サイズ、明るさ、などのウィンドウプロパティを設定可能です。詳細は以下の章を参照してください。

- 2.4 シリアルモジュール (21 ページ)
- 2.5 TCP モジュールとコマンドラインウィンドウの操作 (26 ページ)
- 3 ウォールのプログラミング (WIL) (63 ページ)

2.1.3.最大限のビデオウォール制御

DXInterface はウォール制御に必要な全機能のサブセットを提供しています。
ユーザーがすべての機能を使用する必要がある場合は、独自ソケットライブラリのダイレクトプログラミングが利用可能です。これは TCP での定義済みパケットを使用するバイナリプロトコルです。最新の「DXWallServer Socket Interface」のドキュメントを参照してください。

2.2. フレームワークと基本設定

2.2.1.インストールと設定ディレクトリ

DXInterface のインストールでは、以下の読み取り専用のディレクトリを使用します。

- **アプリケーションファイル**
  C:¥Program Files¥DEXON Systems¥DXWall Control

- **画像ファイル**
  C:¥Program Files¥DEXON Systems¥DXWall Control¥images

- **HTML ドキュメント**
  C:¥Program Files¥DEXON Systems¥DXWall Control¥doc

- **設定用プログラム**
  Start Menu¥DXWallControl¥DXInterface Configuration

- **ドキュメント**
  Start Menu¥DXWallControl¥DXInterface Documentation

DXInterface の構成ファイルおよびディレクトリは以下になります。

- **DXInterface configuration で作られた構成ファイル**
  C:¥ProgramData¥DXInterface¥config

- **ユーザーの構成データファイル**
  C:¥ProgramData¥DXInterface¥data
  *.dxs シナリオファイルと HTML ファイルをここに保存します。

- **DXInterface のログ**
  C:¥ProgramData¥DXInterface¥log

【注意】C:ProgramData は隠しフォルダです。

2.2.2. 設定方法

本機は全ての設定データを設定ファイルに保存します。
これらの設定ファイルはサービスの config ディレクトリ内にあります。各モジュールはそれぞれの設定ファイルを持っています。設定ファイルのフォーマットは XML です。設定ファイルは別の場所に保存することも可能で、後でコピーして戻すこともできます。ファイルの手動での編集はしないでください。全ての設定パラメータは、DXInterface サービスの設定ダイアログで設定できます。
スタートメニューの「DXInterface Configuration」を起動してください。インターフェースサービスが実行されていない場合は、ダイアログが表示されません。サービスが開始途中の場合は、エラーメッセージが表示されます。数秒待ってから、再度メニューをクリックしてください。

設定ウィンドウでは以下のメニューを使用できます。

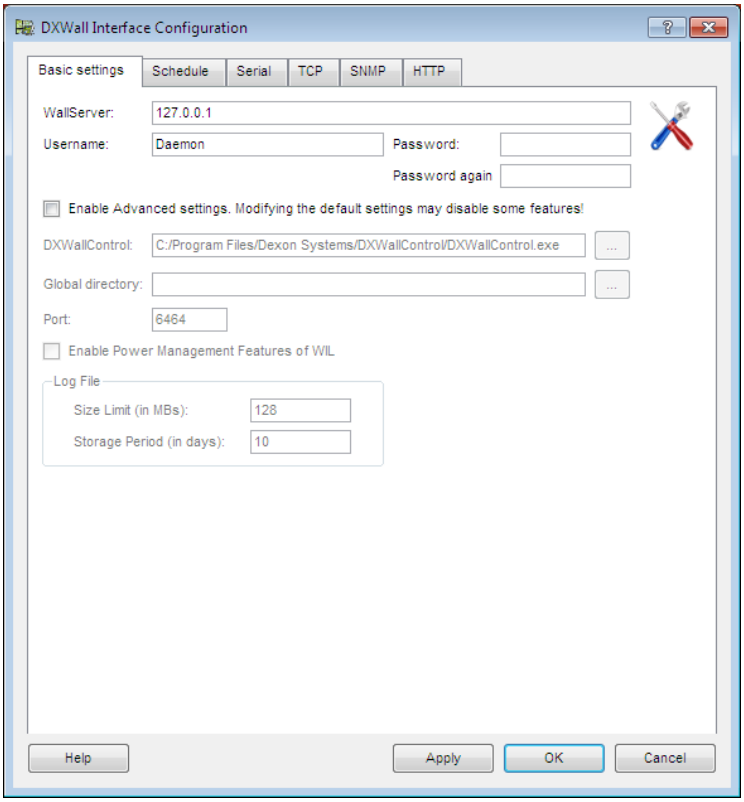
- **Configuration**
  設定ダイアログを起動します。全てのグローバル設定、モジュール固有の設定を変更することができます。
- **Resume**
  有効なモジュールを開始します。
- **Pause**
  全てのモジュールを停止させます。サービス自体は引き続き実行します。
- **Log Types**
  システムが表示するメッセージの種類を設定できます。詳細は次の章を参照してください。
- **Clear logs**
  設定ダイアログに表示されているログを消去します。
- **Export**
  設定内容を任意のディレクトリに保存します。
- **Import**
  保存した設定を読み込むことができます。

DXInterface の設定ダイアログでは、ウィザードを使用してシナリオを簡単に設定することができます。

- **Desktop shortcut wizard**
  定義済みのウィンドウレイアウトやシナリオを開くショートカットをデスクトップやその他の場所に作成できます。このウィザードにはショートカットにキーの組み合わせを割り当てるオプションがあり、キーボードで呼び出すことができます。また、ローカルで使用するための TCP モジュールを設定します。
- **Attach Device wizard**
  TCP ソケット、または RS-232 ポートで通信可能な外部デバイス (AMX、Crestron など) を本機に接続して、定義済みのレイアウトを呼び出すことができます。このウィザードで TCP / シリアルモジュールの設定を行い、本機と外部デバイスの構成を手助けします。
- **HTML Browser control wizard**
  DXInterface に組み込まれたウェブサーバを設定し、ボタン付きの自動生成ウェブページを提供します。このウィザードで HTTP モジュールを設定し、ウェブページのボタンを使用して定義済みのレイアウトを呼び出すことができます。
- **Startup layout wizard**
  本機の起動後に表示されるレイアウトや自動シーケンスを設定できます。ローカルで使用する TCP モジュールを設定します。

2.2.3. 基本設定

Basic settings モジュールでは、DXWallServer サービスとの接続の管理を行います。接続の準備ができていない場合、インターフェースは数秒後に再接続を試み、接続が確立されるまでエラーメッセージを送信します。



1. **Wall Server**
   本機の IP アドレスまたは PC ホスト名です。ローカルで実行する場合は 127.0.0.1 を入力してください。正常に動作させるためには、インターフェースはサーバーへの接続を維持する必要があります。接続が失われた場合は、数秒間隔で接続を再確認します。
   デフォルトは空欄です。この場合、接続先にはローカルホストが選択されます。インターフェースが開始され、モジュールが接続を要求した時に接続が確立されます。
2. **Username**
   接続に使用するユーザー名です。管理者 (Administrator) ユーザーは常に有効です。すべてのクライアントはここで設定したユーザー権限でウォールにアクセスします。
3. **Password**
   ユーザーのパスワードです。デフォルトは空欄です。
4. **Password again**
   確認のためにパスワードの再入力をします。
5. **DXWallControl**
   DXWallControl アプリケーションのパスです。ソフトウェアのインストール時に有効な値が入力されます。**変更しないでください。**
6. **Global directory**
   セキュリティオプションです。インターフェースのマッピングにシナリオファイルが直接指定されていても、全てのモジュールがハードディスクのどの部分にもアクセスできないように設定できます。クライアントはここで指定されたディレクトリにのみアクセス可能になります。空欄の場合は、ハードディスク上のすべての HTML ファイルとシナリオファイルへの読み取りアクセス

を全モジュールに許可します。ソフトウェアのインストール時に有効な値が入力されます。**変更しないでください。**

7. **Port**
   DXWallServer サービスが待ち受けているポートです。デフォルトポート番号は 6464 です。ソフトウェアパッケージのインストール時に有効な値が入力されます。**変更しないでください。**
8. **Enable Power Management Features of WIL**
   チェックを有効にすると、リモートセッションが PWR コマンドを使用して本機のシャットダウンができるようになります。

   【注意】この機能を使用するためには、DXInterface が本機で起動している必要があります。
9. **Log File Size Limit**
   1 日のログファイルのサイズ制限をメガバイト単位で指定できます。ログ情報の量が制限に達した場合、ファイルの古い内容が解除され、ログファイルの先頭からログが継続されます。「0」を入力した場合、ログファイルのサイズ制限が無しになります。
10. **Storage Period**
    ログファイルの保存期間を日単位で指定できます。保存期間が 0 に設定されている場合、自動削除機能は無効になります。

2.2.4.サービスの使用と設定

DXInterface モジュールは、通常サービスとして動作しますが、アプリケーションとしても実行できます。以下のコマンドライン引数が使用できます。

dxinterface.exe [-a | -i | -e | -t | -u | -p | -r | -c 1]

一度に使用できる引数は 1 つだけです。

コマンドライン引数	
-a	この引数を指定すると、プログラムはアプリケーションとして機能します。 ウィンドウの左上の角にある×ボタンで閉じることができます。
-i	現在の実行ファイルをシステムサービスとしてインストールします。 **重要**：ローカルディレクトリパスから実行する必要があります。ネットワークアドレスやマップされたネットワークドライブからの実行はできません。
-e	サービスを開始します。 サービスを -i オプションでインストールした場合、[コントロールパネル] → [管理ツール] → [サービス] にあるサービスダイアログで開始 / 停止が可能です。
-t	サービスのインスタンスを終了させます。 同時にひとつのサービスインスタンスだけ起動できます。このオプションはそれを停止させます。サービスを -i オプションでインストールした場合、[コントロールパネル] → [管理ツール]→ [サービス] にあるサービスダイアログで開始 / 停止が可能です。
-u	サービス一覧からサービスをアンインストールします。サービスがインストールされていない場合は、このオプションは無視されます。
-p	サービスの一時停止をします。 Configuration メニューの中にある Pause メニューを使って同じことができます。全てのモジュールを停止させますが、サービスはアイドル状態で起動を続けます。
-r	サービスを再開させます。 Configuration メニューの中にある Resume メニューを使って同様のことができます。有効な全てのモジュールを開始します。
-client	サービスウィンドウの configuration を起動します。このコマンドはサービスが起動している状態でのみ有効です。スタートメニューにある「DXInterface Configuration」の実行と同様の動作になります。管理者グループのユーザーのみアクセスできます。

2.2.5. ログの生成

ログは外部デバイスやソフトウェアの開発者の助けになります。なお、セキュリティやウォールコントローラーの問題を見つける際にも有効です。インターフェースとデバイスが開発された後の通常使用時は、全てのメッセージを無効にしておくことを推奨します。

ログは通常、デフォルトのディレクトリ内に「logYYYY_MM_DD.log」のファイル名で作られます。YYYY_MM_DD は現在の日付を表します。ファイルサイズは 128MB までに制限されています。一日のうちにファイルサイズの制限を超えてしまった場合は、現在のログファイルが上書きされ、それまでのその日全てのログが消去されます。
10 日前よりも古いログファイルは新しいログが保存された際に自動的に消去されます。ログメッセージは設定ウィンドウのテキスト部分にも送られ、ダイアログが表示中にのみ表示されます。ログファイルはディレクトリ%ALLUSERSPROFILE%/DXInterface/log に保存されます。

ログには以下の 4 つのタイプがあります。

- **Error**：
  最も深刻な問題はここに記録されます。一般にこれらは、設定された機能が通常通り動作せずにシステムを妨げているような問題です。
- **Warning**：
  深刻な問題の場合もありますが、システムモジュールは動作しています。Warning メッセージは設定の中で無効にされているマッピングが呼び出されたような場合に記録されます。
- **Info**：
  正常な動作に関する情報です。
- **Debug**：
  デバッグ機能は設定段階で有用です。一般的のケースでは、デバッグメッセージは全ての受信したバイトデータを記録します。

デバッグのレベルは、メインウィンドウの Configuration メニューで個別に有効 / 無効を設定できます。この設定は設定ファイルに保存されます。

2.3. スケジューラ・モジュール

2.3.1.概要

スケジューラを使用して、作成済みシナリオステップの実行をスケジュールし、作成済みのレイアウトをビデオウォールに表示できます。スケジューラはフレームワーク及びこのモジュールを有効にしたときに毎回起動します。

スケジューラでは以下のことができます。

- レイアウトや自動シーケンスを毎日、毎週、毎月、または特定の時間に実行
- イベントのスケジュールを変更
- スケジュールされたイベントの停止

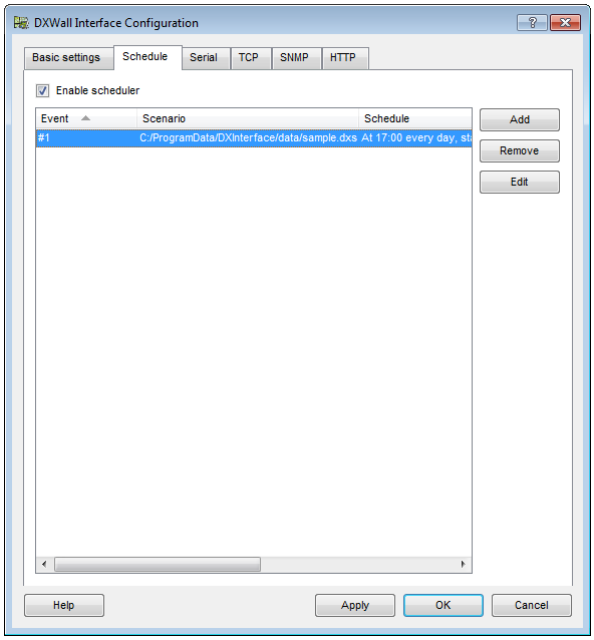
2.3.2.クイックスタート

スケジューラを設定するには、DXWall Interface のメニューから configuration を開きます。

1. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、DXWallControl アプリケーションを閉じます。
2. 「Enable」にチェックを入れてモジュールを有効します。
3. add をクリックし、イベントタイプを「Once」に設定して、現在の時刻より少なくとも 2 分後以降の時刻を設定します。OK を押してください。
4. リストにスケジュールされたイベントが表示され、「Next execution time value」 が有効になっていることを確認します。「OK」または「Apply」を押してください。
5. 設定した時刻にスケジューラが動作し、レイアウトが表示されます。その際、システムログにもメッセージが記録されます。

2.3.3.スケジューラ・エディタ

スタートメニューから [DXWallControl] → [DXWall Interface] を起動し、Configuration を選択します。Scheduler タブを開くと、以下のスケジューラ・エディタのフォームが表示されます。



ダイアログの左側に設定済みのイベントが表示されます。イベントには、名前 (Event) とシナリオファイル (Scenario) が関連付けられています。同じ名前のイベント名で、2 つの異なるシナリオファイルをもつイベントにも対応できます。

- **Schedule**
  現在のスケジュールの説明が表示されます。
- **Next Exec Time**
  次の実行時間を表示します。
- **Last Exec Time**
  前回実行された時間を表示します。
- **Last Execution**
  実際に実行されたイベントを意味します。Last execution の日時は、スケジュールされた時刻になってもイベントが実行されなかった場合には変更されません。一方で、ダイアログが開かれているかどうかに関わらず、イベントが有効であればイベントは実行されます。

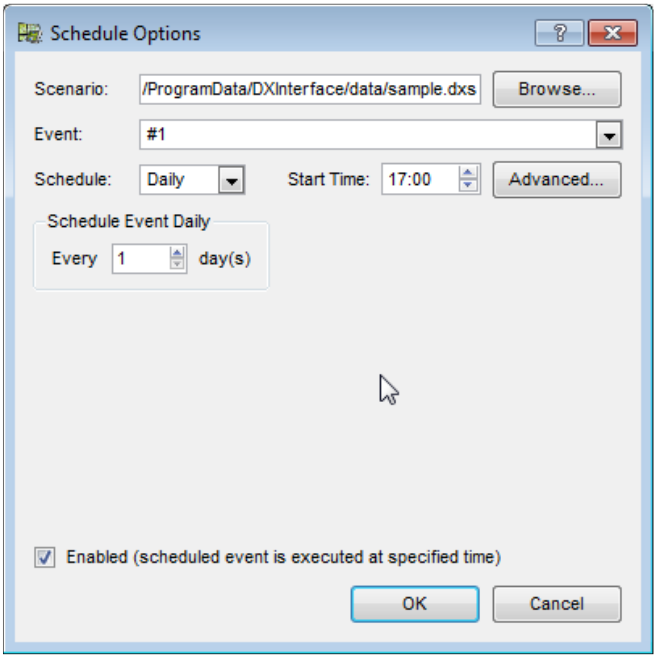
Add ボタンを押すと新しいイベントがスケジュールに追加されます。オプション設定は別のダイアログで設定します。

リスト内のイベントを選択し、Remove または Edit ボタンを押すことで、イベントの修正または削除ができます。

DXWallServer のアドレスなどのイベント実行の設定は、ダイアログの Global タブで設定できます。
OK、Apply、Cancel のいずれかのボタンを押すと、ダイアログが閉じます。

2.3.4.スケジュールオプション

オプションダイアログにて、イベントのスケジュールパラメータを設定できます。新規イベントの設定や既存のイベントの編集にも使われます。



最初にシナリオファイルを選択します。シナリオファイルの選択は 2 通りの方法があります。

- シナリオファイルのパスを **Scenario** の欄に入力する。
- **Browse** ボタンをクリックしてファイルを選択する。

シナリオファイルが選択されると、Event ボックスにそのシナリオのシリアル番号または外部イベントが表示されます。ここからイベントを選択する必要があります。

イベントの選択後、そのイベントを実行する頻度を設定します。 (Once、Daily、Weekly、Monthly) この選択したものに応じたスケジュールコントロールがダイアログに表示されます。

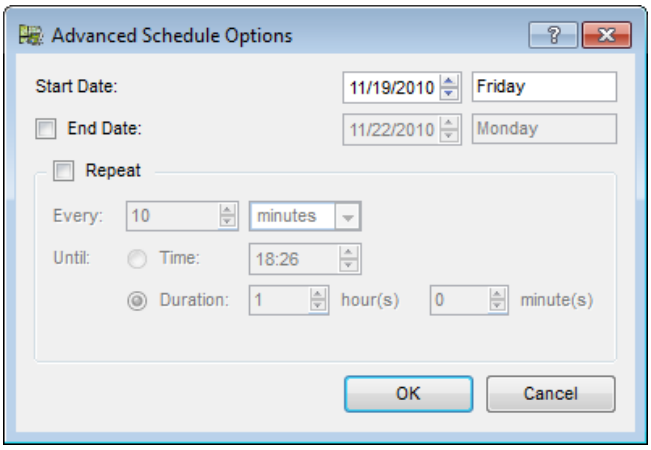
- **Once :**
  実行する日付を設定できます。日付エディタの隣に曜日が自動的に表示されます。
- **Daily :**
  1 日ごとの設定です。イベントを実行する間隔を設定できます。
- **Weekly :**
  1 週間ごとの設定です。イベントを実行する間隔を設定します。1 週間のうち複数の曜日を選択可能です。
- **Monthly :**
  Select Months ダイアログが開き、実行する月を選択できます。日付は 2 つの方法で設定できます。
- **ワンステップでの方法**
  月の中の日付を指定します。もし 29 日から 31 日の間を指定した場合、イベントはその日付が含まれている月でのみ実行されます。
- **ツーステップでの方法**
  最初に月の中にある曜日のうち 1 番目、2 番目、3 番目、4 番目、または最後のイベント実行の日を選択してください。その後、特定の曜日を指定してください。

全てのスケジュールに対してタイムエディタを利用して **Start Time** をセットできます。 Advanced ボタンを押すと、Advanced Schedule Options のダイアログが開きます。

スケジュールごとに Enabled のチェックボックスでイベント実行の有効、無効を設定できます。

2.3.5.拡張オプション (Advanced Schedule Options)

このダイアログではさらに詳細なスケジュールパラメータを設定できます。



- **Start Date**
  イベントの開始日です。選択された日に対応する曜日は自動的に表示されます。
-
- **End Date** (任意)
  イベントの終了日を指定します。もし終了日がセットされていない場合、スケジュールは常に有効となります。

- **Repeat**
  チェックがない場合は、指定された日に一度だけイベントを実行します。チェックを入れた場合は、繰り返しイベントを実行します。「**Every**」で毎分、または毎時間単位で繰り返し間隔を設定します。「**Until**」にある「**Time**」では繰り返しの終了時刻を、また「**Duration**」ではイベントの繰り返しを行う実行時間を設定できます。

2.4. シリアルモジュール

2.4.1.概要とクイックスタート

シリアルモジュールはテキスト、バイナリプロトコル、ユーザー指定のコマンドによる第 1 レベル、第 2 レベルの通信をサポートしています。AMX や Crestron のデバイスに使用することができます。

プロトコル	第 1 レベル (イベント→レイアウト)	第 2 レベル (WIL API)
None	入力コードはユーザーが指定した文字列です。 1 つの入力文字列が入力ストリームで認識されると、新しいレイアウトの表示を実行します。任意の文字が使用可能です。また、表示できない ASCII コードに対しては、¥XX (XX は 16 進数) というコードを使用できます。 例 : ¥0A は LF 、¥5C はバックスラッシュを表します	非対応
Binary	対応	対応
Text	対応	対応

DXInterface には、WIL API と呼ばれるユニバーサルコマンドセットがあります。詳しくは 3. ウォールのプログラミング (WIL) (63 ページ) をご確認ください。

ソフトウェア CD-ROM の中には、シリアルモジュールを学習するためのサンプルビデオファイルがあります。

- Demos¥DXInterface_CreateScenario.wmv
- Demos¥DXInterface_Quickstart_TCP.wmv
- Demos¥DXInterface_HTTP.wmv

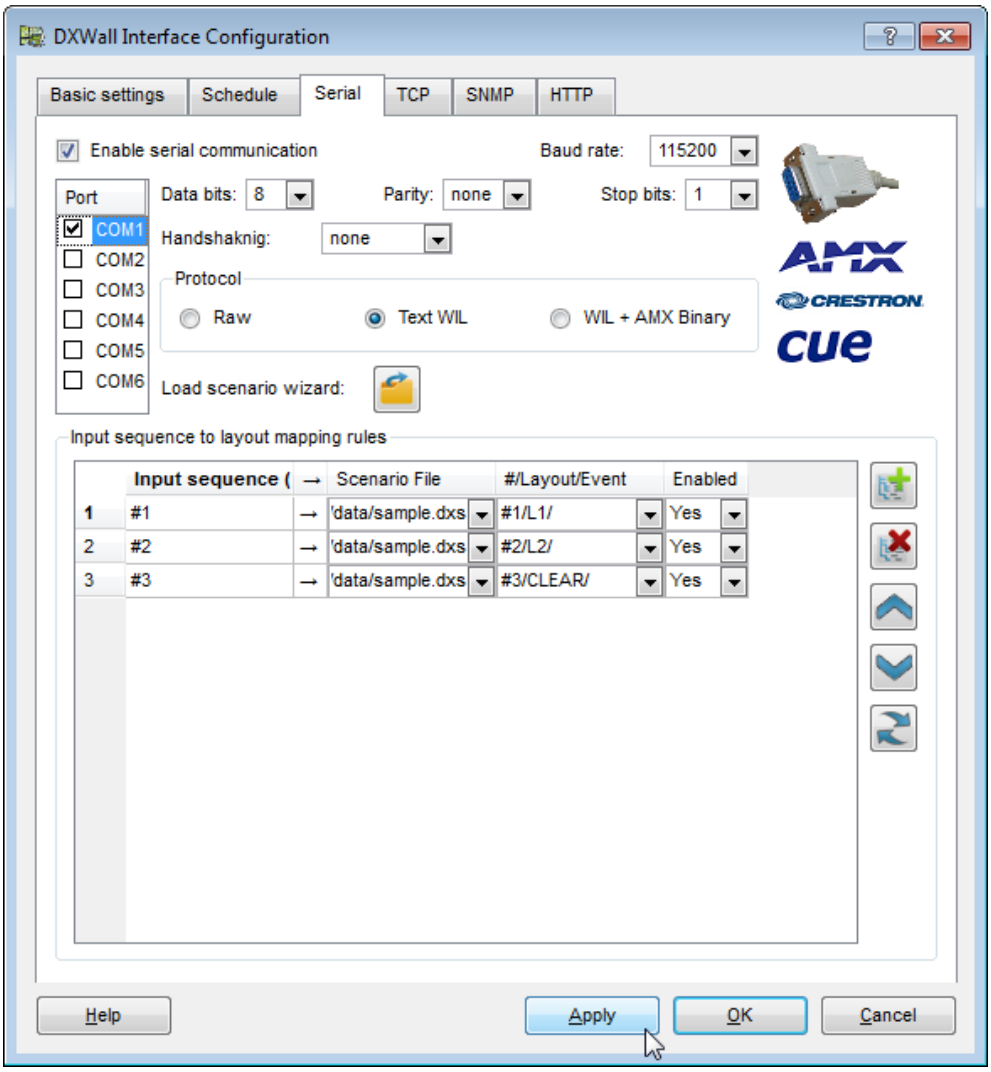
以下では、ユーザー定義のシリアルコマンドでレイアウトを変更する、最もシンプルな例を説明します。Raw プロトコルはこのモジュールを使用する最も簡単な方法です。入力コマンド、シナリオファイル、ステップのペアとのマッピングを定義できます。設定したコマンドが認識されると、シナリオステップが起動します。コマンドに対して 1 つのステップが関連付けされます。コマンドは一意の値でなければなりません。また、ステップは他のステップを含んではいけません。以下のガイドラインに沿って機能をテストします。

1. DXInterface の Configuration ダイアログを開き、Serial タブを選択します。
2. 「Enable Serial Communication」のチェックボックスを選択し、シリアルパラメータを入力します。使用する COM ポートを選択してチェックを入れます。
   今回の例では、ボーレート「115200」、データビット「8」、ストップビット「1」、パリティ「none」、ハンドシェイク「none」にしています。
3. 「Protocol」を「None」に設定します。
4. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、パスを控えておきます。このファイル内のレイアウトは、入力コマンドによって変更されます。
5. シリアルポートからコマンドをキャッチするには、マッピングを作成する必要があります。マッピングでは、シナリオファイルと入力されるイベントコマンドの組み合わせでステップを定義します。Scenario File 欄で作成したシナリオファイルを参照、選択することで、シナリオファイル内のすべてのステップを自動で追加します。読み込みボタン を押すと、Scenario 欄で指定さ

れている現在のシナリオファイルからステップを読み込みます。また、 ボタンを使用することでマッピングを手動入力することも可能です。ステップが選択できない場合は、シナリオのパスが間違っていないか、保存されたシナリオにステップが含まれているかを確認してください。

6. 設定完了後に OK ボタンを押します。設定したシリアルポートに適切なコマンドを送信すると、定義済みのレイアウトが表示されます。

【注意】Protocol 欄に Text WIL を選択している場合、入力コード#1 をイベントとして送信するには、「EVT #1」コマンドの後に CR (Carriage Return), LF (Line Feed) の文字コードを続けて送信する必要があります。



2.4.2. リファレンス

Enable Serial Communication
この項目がチェックされている場合のみ、サービスは指定されたシリアルポートを待ち受けます。チェックしない場合は変更したパラメータは保存されますが、モジュールは動作しません。

Port
使用するシリアルポートを指定します。各シリアルポートにチェックを入れると、そのポートが有効になります。設定とマッピングは、各シリアルポートに関連付けられます。各ポートの設定は、ここにチェックが入っている場合にのみ機能します。別のポートを選択した場合は、そのポートに割り当てられている設定とマッピングをダイアログ上に表示します。
ポートを使用する場合は他のソフトウェアまたはモジュールが同じポートを使用していないことを確認してください。他で使用中に設定を適用した場合、システムがエラーメッセージを表示します。

Baud Rate
使用するボーレートです。低い値から通信テストを実施することをお勧めします。外部のシリアルデバイスと同じ値を設定する必要があります。

Handshaking
ハンドシェイクの種類です。有効値は「None」「Software」「Hardware」です。外部デバイスと同じ値にする必要があります。

Parity
パリティビットの設定です。もしパリティビットを使用しない場合はパリティを偶数、奇数、none のどの値にしても構いません。使用する場合は外部デバイスと同じ値にする必要があります。

Data bits
データビットの数です。外部デバイスと同じ値が使われる必要があります。

Stop bits
ストップビットの長さです。外部デバイスと同じ値を設定する必要があります。

Protocol
通信に使用するプロトコルを選択します。詳細は**プロトコル**の章をご確認ください。

Event mapping rules
DXInterface ソフトウェアは、入力コマンドをシナリオファイル内にあるレイアウトにマップするマッピング構造を採用しています。ステップは外部イベントまたはシリアル番号 (#1 など) によって識別されます。これによりプログラミングの柔軟性を高め、外部デバイスのプログラミングを変更することなくレイアウトの変更が可能になります。

Enabled
イベントの有効 / 無効設定です。No に設定した場合、マッピングは設定ファイルに保存されますが動作はしません。

Input
プロトコルが Raw にセットされている場合、この欄には入力デバイスからのコマンドが入ります。このコマンドを受信した際に、定義済みのレイアウトを表示します。入力値は一意である必要があります。ま

た、それが他の入力文字列に含まれてはいけません。例えば、「Input1」と「Input2」は別のコマンドとして有効な値ですが、「Inp」と「Input」は文字列が重複しているため有効でありません。
プロトコルが Binary もしくは Text に設定されている場合、外部デバイスから指定された EVT コマンドが実行されると、マッピングが有効になります。

Scen.file
使用するシナリオファイルです。ファイルパスを設定します。シナリオファイルの拡張子は「.dxs」です。

#/Layout/Event
シナリオファイル内のレイアウトを識別します。これにはシリアル番号や外部イベントコードが使用されます。コンボボックスからレイアウトを選択します。何も表示されない場合は、シナリオファイルのパスが正しいか確認してください。

マッピングの管理は、フォームの右下にあるボタンで行います。

	リストに現在の値を追加します。テーブル内の項目をクリックすることで、フィールドにそのデータを読み込みます。
	ボタンを押すことで、選択中の項目を削除します。
	隣接する項目の並びを変更します。項目の順序は重要ではありません。
	ボタンを押すことで、コンボボックス内のステップのリストを更新します。
	選択したシナリオファイルからステップをリストにロードします。

2.4.3. トラブルシューティング

接続できない
ハードウェアの接続を確認してください。インターフェースとデバイスの両方が同じ通信設定を使用しているかどうかを確認してください。
機器を再起動してください。サービスが実行されているか、シリアルモジュールが有効になっているか、選択したポートがポートリストで有効になっているかを確認してください。
他のプログラムや DXInterface モジュールがシリアルポートを使用していないか確認してください。

コマンド送信は成功しているが、レイアウトが表示されない
configuration メニューの全てのタイプのログメッセージを有効にしてください。通信エラーのメッセージが表示されている場合は、Cue プログラムを確認してください。それ以外の場合は、マッピングが定義されていて有効になっているか、有効なデータを含んでいるかを確認してください。問題がなければ、configuration ウィンドウの「Global」タブの設定を確認してください。

DXWallServer の問題
本機との接続が問題ないか確認してください。「Global」タブで設定した IP アドレスと TCP ポートの値が正しいことを確認してください。

Binary または Text プロトコルの問題
詳細は 3. ウォールのプログラミング (WIL) (63 ページ) の章を確認してください。

2.5. TCP モジュールとコマンドラインウィンドウの操作

2.5.1.概要とクイックスタート

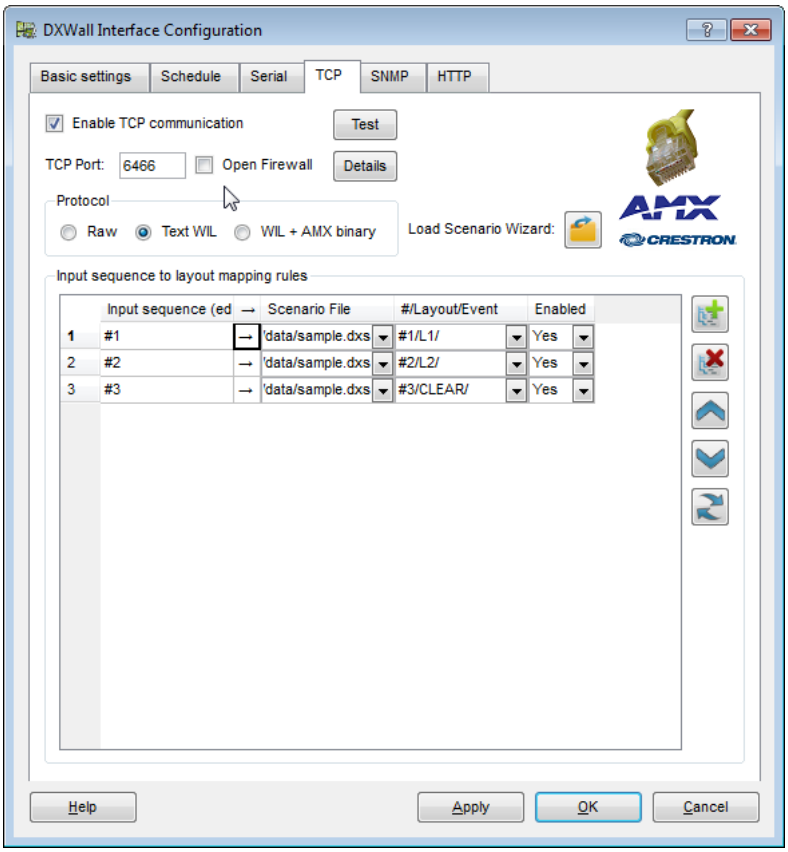
TCP モジュールはシステム内の任意の TCP ポートを待ち受け、ユーザー定義のコマンドの実行、またはユーザー定義の入力コマンドで作成済みレイアウトを表示することが可能です。このモジュールはテキスト、バイナリプロトコル、ユーザー指定のコマンドによる第 1 レベル、第 2 レベルの通信をサポートしています。

プロトコル	第 1 レベル (イベント→レイアウト)	第 2 レベル (WIL API)
None	入力コードはユーザーが指定した文字列です。 1 つの入力文字列が入力ストリームで認識されると、新しいレイアウトの表示を実行します。任意の文字が使用可能です。また、表示できない ASCII コードに対しては、¥XX (XX は 16 進数) というコードを使用できます。 例 : ¥0A は LF 、¥5C はバックスラッシュを表します	非対応
Binary	対応	対応
Text	対応	対応

DXInterface には、WIL API と呼ばれるユニバーサルコマンドセットがあります。詳しくは 3. ウォールのプログラミング (WIL) (63 ページ) をご確認ください。

ソフトウェア CD-ROM の中には、シリアルモジュールを学習するためのサンプルビデオファイルがあります。

- Demos¥DXInterface_CreateScenario.wmv
- Demos¥DXInterface_Quickstart_TCP.wmv
- Demos¥DXInterface_HTTP.wmv
- Demos¥DXInterface_Firewall.wmv



Raw プロトコルのクイックスタート

Raw プロトコルはこのモジュールを使用する最も簡単な方法です。入力文字列、シナリオファイル、ステップのペアとのマッピングを定義できます。設定した入力文字列が認識されると、シナリオステップが起動します。入力文字列に対してひとつのステップがマップされます。入力文字列は一意の値でなければなりません。また、ステップは他のステップを含んではいけません。以下のガイドラインに沿って機能をテストします。

1. DXInterface の Configuration ダイアログを開き、TCP タブを選択します。
2. 「Enable TCP communication」のチェックボックスを選択し、TCP ポート番号を設定します。ここで設定されるポート番号が他のプログラムで使用されていないことを確認してください。
3. プロトコルに「Raw」を設定します。
4. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、そのパスを控えておきます。このファイル内のレイアウトは、有効なコマンドを受信すると表示されるようになります。
5. TCP ソケットでコマンドをキャッチするには、マッピングを作成する必要があります。マッピングでは、シナリオファイルと入力されるイベントコードの組み合わせでステップを定義します。Scenario File 欄で作成したシナリオファイルを参照、選択することで、シナリオファイル内のすべてのステップを自動で追加します読み込みボタン を押すと、Scenario 欄で指定されている現在のシナリオファイルからステップを読み込みます。また、 ボタンを使用することでマッピングを手動入力することも可能です。ステップが選択できない場合は、シナリオのパスが間違っていないか確認してください。
6. OK ボタンを押します。ソケットへの入力コマンドを受信すると、定義済みのレイアウトが表示されます。putty 等の Raw モードのシンプルな Telnet プログラムで確認できます。

【注意】サードパーティ製プログラムの使用は、弊社では責任を負いかねますのでご了承ください。

【注意】Protocol 欄に Text WIL を選択している場合、入力コード#1 をイベントとして送信するには、EVT #1 の後に CR (Carriage Return), LF (Line Feed) の文字コードを続けて送信する必要があります。

Text WIL プロトコルのクイックスタート (イベント→レイアウト機能)
テキストプロトコルを使用してイベントを本機に送信できます。

1. 前述の Raw プロトコルのクイックスタートに従います。
2. Protocol を「Raw」ではなく「Text WIL」を選択します。
3. OK を押します。
4. 本機にて dxlink コンソールを開きます。dxlink 127.0.0.1 –p 6466
5. 以下のコマンドを入力し Enter を押します。EVT STEP1

【注意】STEP1 はシナリオファイルから読み込まれたマッピングの入力コードでなければなりません。

Text WIL プロトコルのクイックスタート (コマンド送信)
このプロトコルでは、カスタムスクリプトを使用して、コマンドライン入力からスクリプトを呼び出すことができます。%PATH%環境変数に追加される DXWallControl のインストールディレクトリ内にある dxlink アプリケーションを使用します。

dxlink [<address>] [-o] [-p <port>] [-c <command>]

コンソール入力またはファイルから本機にファイルを送信します。

- **<address>**
  DXInterface サービスが起動しているコンピュータの IP アドレスです。パラメータ無しで dxlink.exe を実行した場合は、自動的にローカルホスト：6466、ローカルホスト：6465 に接続します。
- **-p**
  DXInterface サービスの TCP モジュールが待ち受けるポートです。デフォルトの値は：6466 です。
- **-o**
  標準入力を標準出力に書き込み可能にするオプションスイッチです。アプリケーションは標準入力からの行をサーバーに転送します。サーバーからの応答は標準出力にコピーされます。
- **-c <command>**
  指定した <command> を TCP で送信し QUIT を呼び出すオプションスイッチです。指定がない場合は、コマンドは標準入力からコピーされます。

【注意】このオプションは dxlink バージョン 1.1.2 以降で動作します。

DXInterface の TCP テキストモードは、以下の方法で構成できます。

1. DXInterface の Configuration ダイアログを開き、TCP タブを選択します。
2. 「Enable TCP communication」のチェックボックスを選択し、TCP のポート番号を設定します。このポートが他のプログラムで使用されていないことを確認してください。
3. Protocol を「Text WIL」に設定します。

Text の例1：**dxlink 127.0.0.1 -o -p 6466 -c "EVT EVT1"**
DXInterface TCP モジュールが EVT1 を受信できるように設定しておきます。イベント (EVT1) を送信して、DXWallControl シナリオステップで定義されたレイアウトを表示するコマンドを実行します。

Text の例 2：**dxlink 127.0.0.1 -o -p 6466 <script.txt**
ビデオソースを開く WIL コマンドを含んだスクリプトを実行します。
以下もご参照ください。

- 4. APPENDIX. チュートリアル (100 ページ)
- 3. ウォールのプログラミング (WIL) (63 ページ)

Windows コンソールを実行し、以下のコマンドを入力してください。

```
dxlink 127.0.0.1 –o –p 6466 <script.txt
```

script.txt の内容：

```
OPN VIDEO1
SET RESULT $VIDEO1
END
```

Text の例 3：dxlink 127.0.0.1 -o -p 6466
上記コマンドを入力した後に、各コマンドを手動で入力することも可能です。

2.5.2. リファレンス

Enable TCP Communication
この項目がチェックされている場合のみ、サービスは指定された TCP ポートを待ち受けます。チェックしない場合は変更したパラメータは保存されますが、モジュールは動作しません。

TCP port
待ち受けに使用する TCP ポートです。

Open Firewall / Details
Windows ファイアウォールでリモート接続を有効または無効にすることができます。また、リモートホスト上のページにアクセスできるようにファイアウォールの設定を変更することができます。「Details」ボタンを押すと、システムのファイアウォール設定を確認・更新できます。

Protocol
通信に使用するプロトコルを選択します。詳細は**プロトコル**の章をご確認ください。

Event mapping rules
DXInterface ソフトウェアは、入力コマンドをシナリオファイル内にあるレイアウトにマップするマッピング構造を採用しています。ステップは外部イベントまたはシリアル番号 (#1 など) によって識別されます。これによりプログラミングの柔軟性を高め、外部デバイスのプログラミングを変更することなくレイアウトの変更が可能になります。

Enabled
イベントの有効 / 無効設定です。No に設定した場合、マッピングは設定ファイルに保存されますが動作はしません。

Input
プロトコルが Raw にセットされている場合、この欄には入力デバイスからのコマンドが入ります。このコマンドを受信した際に、定義済みのレイアウトを表示します。入力値は一意である必要があります。また、それが他の入力文字列に含まれてはいけません。例えば、「Input1」と「Input2」は別のコマンドとして有効な値ですが、「Inp」と「Input」は文字列が重複しているため有効でありません。
プロトコルが Binary もしくは Text に設定されている場合、外部デバイスから指定された EVT コマンドが実行されると、マッピングが有効になります。

Scen.file
使用するシナリオファイルです。ファイルパスを設定します。シナリオファイルの拡張子は「.dxs」です。

#/Layout/Event
シナリオファイル内のレイアウトを識別します。これにはシリアル番号や外部イベントコードが使用されます。コンボボックスからレイアウトを選択します。何も表示されない場合は、シナリオファイルのパスが正しいか確認してください。

マッピングの管理は、フォームの右下にあるボタンで行います。

	リストに現在の値を追加します。テーブル内の項目をクリックすることで、フィールドにそのデータを読み込みます。

	ボタンを押すことで、選択中の項目を削除します。
	隣接する項目の並びを変更します。項目の順序は重要ではありません。
	ボタンを押すことで、コンボボックス内のステップのリストを更新します。
	選択したシナリオファイルからステップをリストにロードします。

2.5.3. トラブルシューティング

接続できない
他のプログラムや DXInterface モジュールが TCP ポートを使用していないか確認してください。
DXInterface のログ出力が問題の発見に役立つ場合があります。

コマンド送信は成功しているが、レイアウトが表示されない
configuration メニューの全てのタイプのログメッセージを有効にしてください。通信エラーのメッセージが表示されている場合は、Cue プログラムを確認してください。それ以外の場合は、マッピングが定義されていて有効になっているか、有効なデータを含んでいるかを確認してください。問題がなければ、configuration ウィンドウの「Global」タブの設定を確認してください。

DXWallServer の問題
本機との接続が問題ないか確認してください。「Global」タブで設定した IP アドレスと TCP ポートの値が正しいことを確認してください。

Binary または Text プロトコルの問題
詳細は 3. ウォールのプログラミング (WIL) (63 ページ) の章を確認してください。

2.6. AMX デバイスでの使用

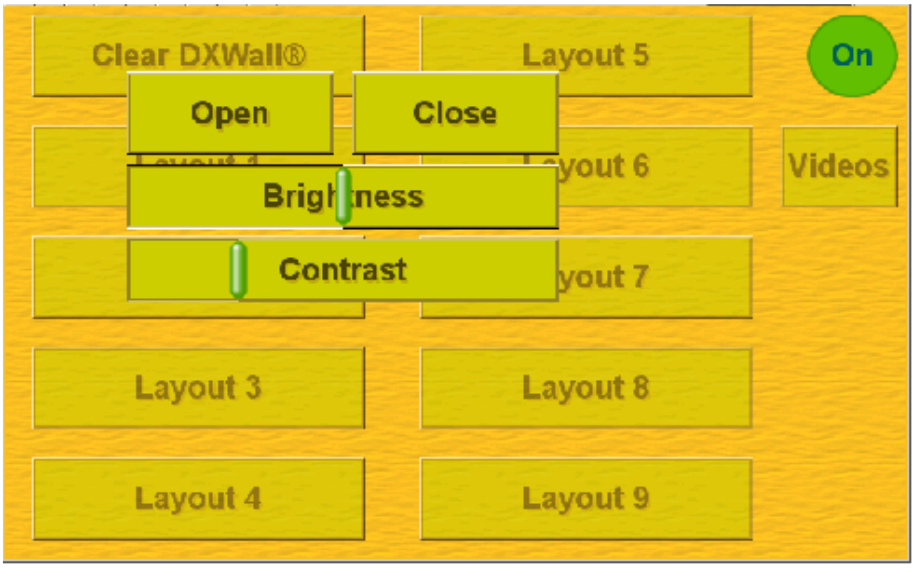
DXInterface の TCP / シリアルモジュールは、AMX Netlinx デバイスとの通信機能を備えています。TCP / IP 接続を使用する場合、DXInterface は TCP サーバーとして動作します。通信は全二重通信ですが、ほとんどがプロセッサによって制御されます。すべての機能を利用するには、通信にバイナリまたはテキストの WIL プロトコルを使用する必要があります。プロトコルについては本紙の appendix の章に記載しています。サンプルでは、NI-700 コントローラと 7 インチのタッチパネル用に作成されています。サンプルには、NetLinx アプリケーション、プロトコルを実装したモジュール、タッチパネルファイルが含まれています。これらはすべてソースコードでアクセスできます。

2.6.1.概要とクイックスタート

モジュールの機能を確認するために、タッチパネルとプロセッサのファイルを作成し、それらをデバイスにダウンロードします。すべてのファイルは CD-ROM 内の Manuals/DXInterface/samples/amx ディレクトリにあります。NetLinx Studio は、2.2.0.84 以降のバージョンを使用してください。テストパネルには、イベント入力やイベントペアを作成するボタンがいくつかあります。

DXInterface の configuration ダイアログを開きます。

1. 「Global Settings」タブにてサーバーの IP アドレスを設定します。
2. 「Serial」タブを選択します。
3. 「Enable Serial Communication」のチェックボックスを選択し、シリアルパラメータを入力します。使用する COM ポートを選択してチェックを入れます。今回の例では、ボーレート「115200」、データビット「8」、ストップビット「1」、パリティ「none」、ハンドシェイク「none」にしています。
4. Protocol を「WIL ＋ AMX Binary」に設定します。
5. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、そのパスを控えておきます。このファイル内のレイアウトは、有効なコマンドを受信すると表示されるようになります。
6. コマンドをキャッチするには、マッピングを作成する必要があります。マッピングでは、シナリオファイルと入力されるイベントコードの組み合わせでステップを定義します。Scenario File 欄で作成したシナリオファイルを参照、選択することで、シナリオファイル内のすべてのステップを自動で追加します。読み込みボタン  を押すと、Scenario 欄で指定されている現在のシナリオファイルからステップを読み込みます。また、  ボタンを使用することでマッピングを手動入力することも可能です。
7. OK ボタンを押します。
8. レイアウトを表示するために、 EVT <input sequence> WIL メッセージを送信するよう AMX デバイスを設定します。CLR コマンドを送信するとウォール上のウィンドウがすべて閉じられます。このサンプルでは、10 進数の 0、1、2、3、4、5、6、7、8、9 が有効な入力イベントになります。TCP ではなく、シリアル通信を使用するようにサンプルを更新します。これについては、後述の章で説明します。



イベントを送信すると、各マッピングによって定義されたレイアウトが表示されます。「Clear Wall」ボタンは「CLR」の WIL コマンドを送信します。「Videos」ボタンは、ソース「VIDEO1」の開閉ができるポップアップを表示します。ソースは DXWall Configurator にて名称を「VIDEO1」に設定する必要があります。また、このサンプルではウィンドウの明るさとコントラストの初期値をポーリングし、ユーザーがそれらを設定できるようになっています。詳細については、ウォールのプログラミング (WIL) の章をご確認ください。

2.6.2. プロセッサのプログラミング

AMX デバイスと DXInterface server 間のプロトコルはシンプルで、プロトコルメッセージは AMX デバイス標準の Marshaling Protocol を使って生成されます。できる限り簡単に使用するために、このプロトコルは NetLinx プログラミング言語の独立したモジュールで実行されます。モジュールは plugins/DXAMOD.AXS と plugins/DXAPROTOCOL.AXI ファイルに含まれています。このモジュールを使用するサンプルは DEMOMAIN.AXS ファイルで実行されます。詳細については**プロトコル**及び **AMX サンプル**に関する appendix をご覧ください。モジュールはウォールの処理を簡略化し、DXWall に依存するコードと依存しないコードを簡単に分離できますが、パフォーマンス上の理由から単一のアプリケーションファイルに書き換えることが可能です。

モジュールはデフォルトで TCP 接続を使用しますが、DEMOMAIN.AXS の一行をコメントアウトし、シリアルパラメータの部分をアンコメントすることで、接続を変更することが可能です。

本システムは通常、初期状態から起動します。サンプルは DEFINE_START セクションの起動後、4 秒間シリアル接続の確立をするようにプログラムされています。

1. DXCONNECT() 関数を呼び出すことで接続を確立し、TCP またはシリアルでの通信が可能な状態になります。
2. DXCONNECT() の呼び出しが繰り返すことで、接続タイプ、接続パラメータ (ボーレート、TCP ポート等) の変更が可能です。
3. モジュールは接続後に PING メッセージをサーバーに送信します。サーバーが応答するとシステムはその状態のままでいます。PING メッセージまたは返信メッセージが失われ、有効な応答が 1 秒以内に返ってこない場合は、「TCP lost」または「Serial lost」状態になります。

4. 本システムは PING メッセージをサーバーに定期的に送信するウォッチドッグ機能を有しています。特定の時間が経過した後にサーバーからの応答がない場合、接続はオフラインであるとみなされます。この場合、OFFLINE のメッセージがアプリケーションに送信され、モジュールは接続の再確立を定期的に試みます。
5. DXDISCONNECT()の呼び出しはウォッチドッグを停止し、どの接続状態であっても接続を終了して、システムに Unknown 状態を送信します。
6. サーバーは任意のタイミングで PING メッセージを送信することも可能です。この場合は、プロセッサ内のモジュールが PONG メッセージで応答します。
7. クライアントは EVT や EVP などのリクエストや、その他の WIL メッセージを送信できます。サーバーはそれらに対して ACK メッセージで応答します。

仮想デバイスを介してモジュールと通信することが可能です。詳細についてはサンプルコードを参照してください。仮想デバイスはモジュールとメインアプリケーション間の接続を行います。仮想デバイスの SEND_COMMAND コマンドを使用して、サーバーへの接続、またはメッセージの送信をできます。モジュールは WIL メッセージと仮想デバイス間の変換を実行します。WIL メッセージの詳細については、ウォールのプログラミング (WIL) の章をご確認ください。
仮想デバイスの拡張機能と制限事項は以下の通りです。

CONFIG [SER, <serialParams>|TCP, <address>, <port>]

このコマンドは接続を確立し、DXCONNECT() 関数を呼び出します。成功すると仮想デバイス経由で ONLINE メッセージを送信します。

- **<serialParams>**
  NetLinx デバイスの「SET BAUD」コマンドに送信されるものと同じシリアルパラメータです。
- **<address>**
  DXInterface サービスが起動しているコンピュータの IP アドレスです。
- **<port>**
  AMX の設定フォームの「TCP port」に設定してあるポート番号です。

サンプル：

CONFIG SER, 115200, N, 8, 1 485 DISABLE
CONFIG TCP, 192.168.1.100, 6465

DISCONNECT
サーバーとの接続を解除します。すべての TCP ソケットを閉じて、待ち受けを終了します。実行完了後に、仮想デバイスを介して OFFLINE メッセージを返信します。

ONLINE
プロセッサがサーバーから DISC メッセージ以外のメッセージを受信した際に、仮想デバイスがこのメッセージを送信します。ONLINE と OFFLINE のメッセージは、接続状態を表示するボタンのステータス更新に使用できます。

OFFLINE
以下の状況の際に、仮想デバイスはこのメッセージを送信します。

1. 仮想デバイスの CONFIG リクエストが成功しなかった場合
2. 仮想デバイスの DISCONNECT リクエストが呼び出され、接続タイプが設定された場合
3. DISC メッセージをサーバーから受信した場合
4. PING メッセージを受信後、PONG メッセージがサーバーから到着しなかった場合
5. シリアルデバイスまたは TCP デバイスのいずれかで OFFLINE データイベントが発生した場合

ONLINE と OFFLINE のメッセージは、接続状態を表示するボタンのステータス更新に使用できます。

ACK <code>
PING と PONG メッセージを除くサーバーへの全てのリクエストは、ACK メッセージで応答します。
ACK メッセージはモジュールを経由して、仮想デバイスの ACK メッセージに変換されます。
<code>パラメータを持つ仮想デバイスの ACK メッセージは以下の通りです。

1. **ACK OK**
   サーバーはリクエストを受け入れ、そのリクエストを実行します。
2. **ACK ERR**
   サーバーはリクエストを理由なく拒否しました。
3. **ACK CHKSUM**
   メッセージ内に checksum エラーがあったため、サーバーはリクエストを拒否しました。

ERR<error string>
モジュールにもエラーが発生する可能性があります。この場合、モジュールは仮想デバイスを介してエラーメッセージを返送します。

<error string>
詳細についてはモジュールのソースコードを確認してください。

INT <value> <serial>
サーバーが OPN によるウィンドウ操作や GET コマンドに対する整数など、何らかの値をクライアントに返信する際に、このコマンドで返送されます。値は 10 進数の整数です。シリアルの値は、GET メッセージによって与えられているシリアル ID を表す文字列値です。

Other WIL commands
クライアントは他のメッセージを送信することもできます。3 つのキーワードで構成される WIL メッセージは、すべてサーバーに送信することができます。

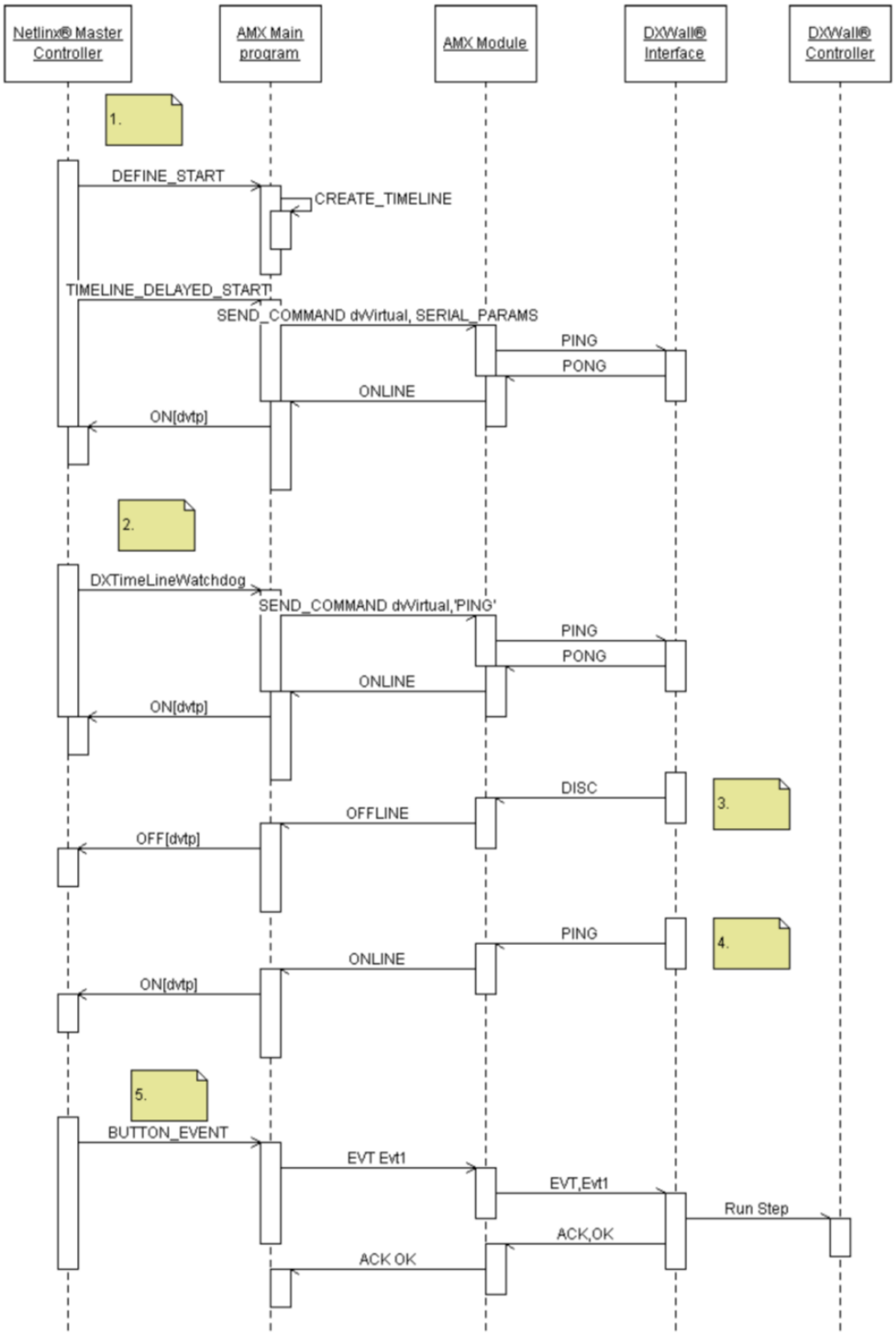
2.6.3. AMX サンプルの使用方法

本機に AMX デバイスを接続して使用する場合の NetLinx サンプルについて説明します。
サンプルには以下 3 つのファイルが含まれています。

1. DXAMOD.DXS
   プロトコルを実装したモジュールです。このモジュールは仮想デバイスをエクスポートします。モジュールとメインプログラム間の通信は、この仮想デバイスを介して行われます。以降はこのファイルをモジュールと呼びます。
2. DXAPROTCOL.AXI
   グローバル定数を含んだファイルです。
3. DEMOMAIN.AXS
   通信のメインプログラムのサンプルです。以降はこのファイルをメインプログラムと呼びます。
4. AMX-DXWall Test.TP4
   TPDesign 4 Touch panel のサンプルファイルです。

以下のシーケンス図では、サンプルの通信の仕組みを説明しています。

1. NetLinx プロセッサが起動し、DEFINE_START シーケンスを呼び出します。しばらくした後に、タイムラインイベントを使用して接続を確立します。
2. ウォッチドッグ・タイマーが動作します。PONG を送信すると、サーバーがその 1 秒後に応答します。
3. サーバーが DISC メッセージを送信して接続を切断し、NetLinx プロセッサはタッチパネルボタンを更新してオフライン状態を表示します。
4. サーバーが再び利用可能になると PING を送信して AMX プロセッサに通知し、プロセッサはステータスボタンをオンラインに更新します。
5. ユーザーがタッチパネルのボタンを押すと、NetLinx プロセッサはイベントコードでモジュールを呼び出し、モジュールは EVT メッセージをサーバーに送信し、サーバーはレイアウトを表示します。



機能の確認のため、この 3 つのファイルのプロジェクトを作成します。

- 最初のステップとして、メインプログラムの最初にある device:port:system の組み合わせを、システムのローカルデバイスと一致するように変更します。
- イベント ID-s の TimeLine 固有の動作で問題が発生する場合があります。モジュール内には、ウォッチドッグ・タイマー用の「DXTimeLineWatchdog」と PING タイマー用の 「DXTimeLineRequest」という 2 つの定数があります。必要に応じてこれらを変更します。メインプログラムにも「TIMELINE_DELAYED_START」が定義されており、これはタッチパネルのボタンがプロセッサの変数と一致するように、起動時の自動接続を遅らせます。DEFINE_START セクションの

DXTimeArray[1] で定義されたウォッチドッグの間隔は 5000 ミリ秒です。PING のタイムアウトは 1000 ミリ秒で、DXPING() 関数の DXTimeArrayRequest で定義されています。

- メインプログラム内の対応する行をアンコメントすることで、TCP とシリアルの接続を切り替えることができます。
- シリアルポートのパラメータはメインプログラムの SERIAL_PARAMS 定数で文字列として定義されます。ハンドシェイクは、サンプルでは常に「なし (none) 」に設定されています。先頭の文字列「CONFIG SER」はコマンドデータを意味し、それ以降の文字列はボーレート等の設定値になります。

モジュールにはいくつかの機能が含まれています。詳細なパラメータの説明は、ソースコードに記載されています。

- COUNT_CHECKSUM
  出力バッファのチェックサム値を推定します。
- DXCONNECT
  物理的な接続を確立します。
- DXDISCONNECT
  システムを Unknown 状態にします。
- DXPING
  PING メッセージを送信してタイマーを起動します。
- DXPONG
  PING への応答が到着したときに呼び出されます。
- DXSENDMESSAGE
  送信メッセージを解析、構築するために呼び出されます。
- DXRECEIVED
  入力チャンネルに文字列が到着した場合に呼び出されます。

2.6.4. トラブルシューティング

サンプルがコンパイルしない、またはシステム上で動かない
サンプルはデモンストレーション用です。サンプルを使うためには AMX プログラマーの経験が必要です。サンプルを自分のシステムに合わせて書き換える必要があります。

接続できない
ハードウェアの接続を確認してください。インターフェースと AMX デバイスの両方が同じ通信設定を使用しているかどうかを確認してください。
AMX デバイスを再起動してください。サービスが実行されているか、モジュールが有効になっているかどうかを確認してください。
他のプログラムや DXInterface モジュールがシリアルポートや TCP ポートを使っていないか確認してください。DXInterface のログ出力が問題の発見に役立つ場合があります。

DXInterface サービスに簡単なイベントを送信しているが、レイアウトが表示されない
configuration メニューの全てのタイプのログメッセージを有効にしてください。通信エラーのメッセージが表示されている場合は、NetLinx プログラムを確認してください。それ以外の場合は、マッピングが定義されていて有効になっているか、有効なデータを含んでいるかを確認してください。問題がなければ、configuration ウィンドウの「Global」タブの設定を確認してください。

DXWallServer の問題
本機との接続が問題ないか確認してください。「Global」タブで設定した IP アドレスと TCP ポートの値が正しいことを確認してください。

Binary または Text プロトコルの問題
詳細は 3. ウォールのプログラミング (WIL) (63 ページ) の章を確認してください。

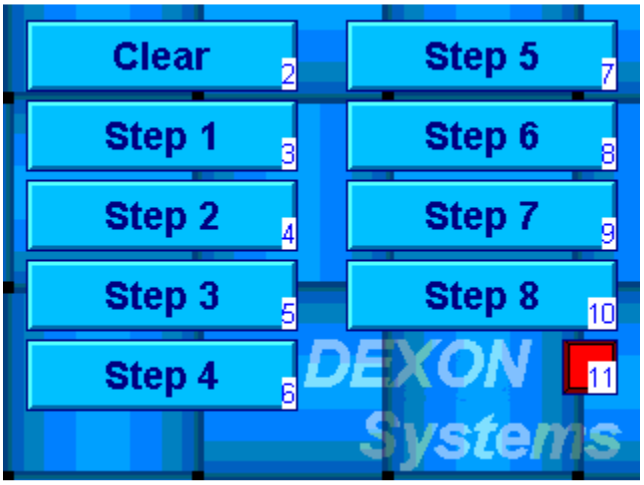
2.7. Crestron デバイスでの使用

DXInterface の TCP / シリアルモジュールは、Crestron デバイスとの通信機能を備えています。TCP / IP 接続を使用する場合、DXInterface は TCP サーバーとして動作し、Crestron デバイスは TCP クライアントとなります。通信は双方向ですが、ほとんどがプロセッサによって制御されます。通信は、特別なテキストコマンドプロトコル (WIL) によって実行されます。
MP2E コントローラと CT-1550 タッチパネルのサンプルを作成しています。TCP 接続用とシリアル接続用の 2 つの SIMPL Windows サンプルファイルがあり、タッチパネル用のファイルも含まれています。また、TCP / シリアル通信による第 1 レベル (コマンドでレイアウト変更) および第 2 レベル (プロトコルを使用してウィンドウを管理) の通信のサンプルも含まれています。これらのサンプルはシンプルな構成で、MP2E プロセッサの I/O ポートを入力として使用しています。すべてのサンプルはソースコード形式でアクセスできます。

2.7.1.クイックスタート

モジュールの機能を確認するために、タッチパネルとプロセッサのファイルを作成し、それらをデバイスにダウンロードします。すべてのファイルは CD-ROM 内の Manuals/DXInterface/samples/crestron ディレクトリにあります。これらのファイルは Symbol and Lib バージョン 232、Crestron DB バージョン 15.9.9、SIMPL バージョン 2.04.11 によって作成されています。個別のイベントを起動するいくつかのボタンが含まれています。
DXInterface の Configuration ダイアログを起動し、Crestron タブを選択します。

1. 「Global Settings」タブにてサーバーの IP アドレスを設定します。
2. 「Serial」タブを選択します。
3. 「Enable Serial Communication」のチェックボックスを選択し、シリアルパラメータを入力します。使用する COM ポートを選択してチェックを入れます。今回の例では、ボーレート「115200」、データビット「8」、ストップビット「1」、パリティ「none」、ハンドシェイク「none」にしています。
4. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、そのパスを控えておきます。このファイル内のレイアウトは、有効なコマンドを受信すると表示されるようになります。
5. コマンドをキャッチするには、マッピングを作成する必要があります。マッピングでは、シナリオファイルと入力されるイベントコードの組み合わせでステップを定義します。マッピングを追加するには、Enabled を有効にし、入力イベントの名前を設定して、シナリオファイルを選択します。Event ボックスで必要なステップのシリアル番号または外部イベントコードを選択します。レイアウトが選択できない場合は、シナリオのパスが正しいことを確認してください。このサンプルで有効な入力イベントは、大文字小文字を区別する ASCII 文字列 EVT0, EVT1, EVT2, EVT3, EVT4, EVT5, EVT6, EVT7, EVT8 です。
6. OK ボタンを押します。接続の確立に関するステータスが表示されます。タッチパネル上の右下にある小さなボタンが、接続が有効な場合は緑、接続を確立できなかった場合は赤になります。



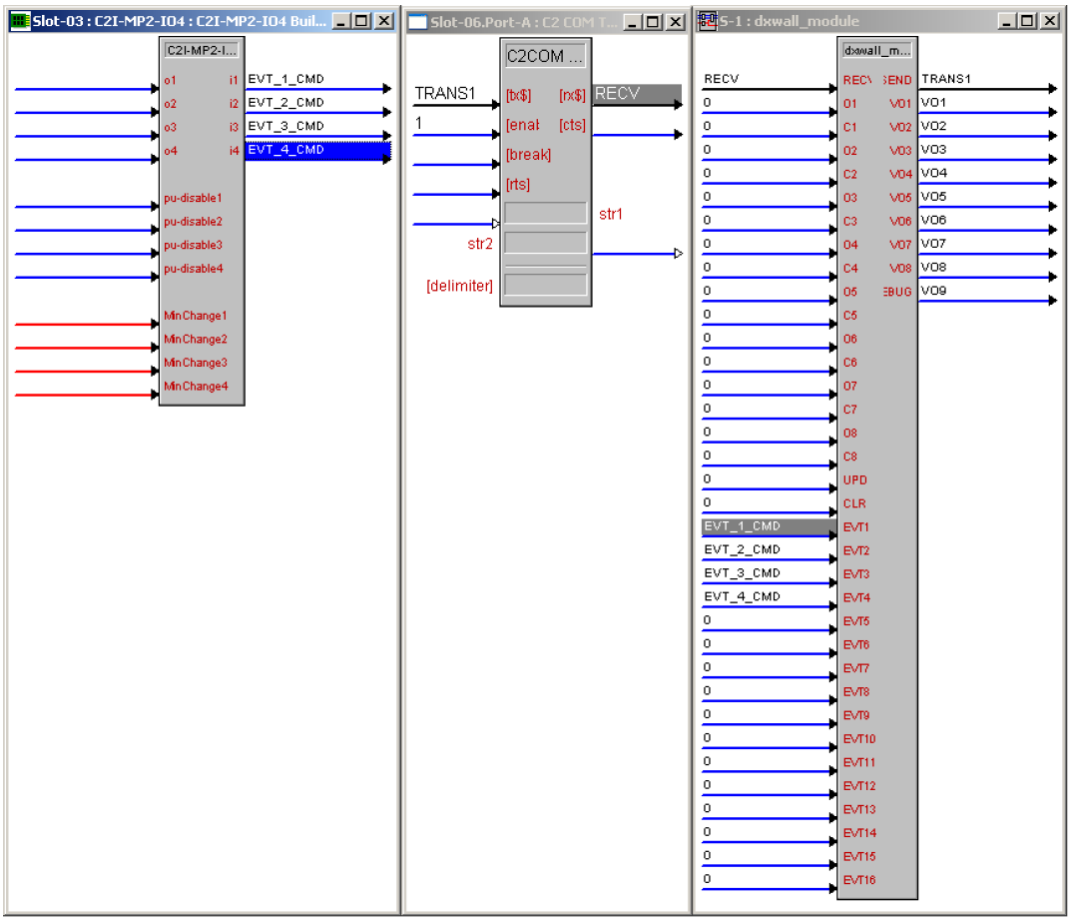
イベントを送信すると、各マッピングによって定義されたレイアウトが表示されます。Crestron システムは、イベントペアを送信することもできます。この場合、シナリオファイルのパスとレイアウトの識別子は、SIMPL プログラムによって直接定義されます。サンプルの「join numbers」は、上の図のようになっています。

2.7.2. プロセッサのプログラミングとサンプル

第 1 レベルのプログラミング

AMX デバイスと DXInterface server 間のプロトコルはシンプルで、プロトコルメッセージはバイナリ形式を使用します。できる限り簡単に使用するために、このプロトコルは 2 つの SIMPL：dxinterface.smw および dxinterface_tcp.smw ファイルで実行されます。

以下の図は、シリアルプログラム「dxinterface.smw」のブロック図です。



本システムは通常、初期状態から起動します。サンプルでは、オシレーターを使用して 4 秒間隔でシリアル接続のチェックを行います。

1. ボタンを押すと、EVT メッセージがサーバーに送信されます。
2. 通信タイプは、DXInterface と Crestron デバイスの両方で設定する必要があります。
3. ウォッチドッグがサーバーへの PING メッセージを開始します。
4. PONG 応答がモジュールの次のタイムアウトまでに返ってきた場合は、PING_TIMEOUT を出力とする D フリップフロップをリセットします。この場合、タイムアウトが発生した次回までに、ステータスボタンが 1 に設定されます。ONLINE2 を出力する D フリップフロップを参照してください。任意の ACK メッセージも同様に動作します。
5. PONG 応答が返ってこない場合は、クロックとなるウォッチドッグによってフリップフロップが「1」に設定されます。この場合、またはサーバーから DISC を受信した場合は、次のタイムアウト (ONLINE2) までにパネルのステータスボタンが赤色になります。
6. サーバーから PING が要求されると、モジュールはその都度 PONG メッセージを送信します。

TCP プログラムは、フリップフロップやウォッチドッグを使用しないという違いはあるものの、他と同じように動作します。この場合、TCP モジュールの Connect-F 信号は、ステータスボタンに直接接続されます。dxinterface_panel.vtp というサンプル用のタッチパネルファイルが用意されています。それを使用してテストを行ってください。

第 2 レベルのプログラミング
Crestron の第 2 レベルのプログラミングでは、8 つのウィンドウを個別に開閉し、必要に応じてイベントメッセージを送信することができます。このサンプルには、モジュール dxwall_module.umc が含まれています。このモジュールには O<X> C<X> 入力があり、デフォルトのパラメータで X 番のウィンドウを開きます。VO<x> 出力は、これらのウィンドウの現在の状態を示します。これらのウィンドウの状態は、クライアントが UPD メッセージを送信すると更新されます。また、イベントコードを生成する EVT イベントや、ウォールをクリアする CLR イベントも受け付けます。サンプルプログラム dxwall_tcp_2.smw は、このモジュールを TCP 経由で使用します。このプログラムは非常にシンプルで、TCP の代わりにシリアル通信を使用するように書き換えることが可能です。

2.7.3. トラブルシューティング

サンプルがコンパイルしない、またはシステム上で動かない
サンプルはデモンストレーション用です。サンプルを使うためには AMX プログラマーの経験が必要です。サンプルを自分のシステムに合わせて書き換える必要があります。

接続できない
ハードウェアの接続を確認してください。インターフェースと Crestron デバイスの両方が同じ通信設定を使用しているかどうかを確認してください。
Crestron デバイスを再起動してください。サービスが実行されているか、モジュールが有効になっているかどうかを確認してください。
他のプログラムや DXInterface モジュールがシリアルポートや TCP ポートを使っていないか確認してください。DXInterface のログ出力が問題の発見に役立つ場合があります。

DXInterface サービスに簡単なイベントを送信しているが、レイアウトが表示されない
configuration メニューの全てのタイプのログメッセージを有効にしてください。通信エラーのメッセージが表示されている場合は、SIMPL プログラムを確認してください。それ以外の場合は、マッピングが定義されていて有効になっているか、有効なデータを含んでいるかを確認してください。問題がなければ、configuration ウィンドウの「Global」タブの設定を確認してください。

DXWallServer の問題
本機との接続が問題ないか確認してください。「Global」タブで設定した IP アドレスと TCP ポートの値が正しいことを確認してください。

Binary または Text プロトコルの問題
詳細は 3. ウォールのプログラミング (WIL) (63 ページ) の章を確認してください。

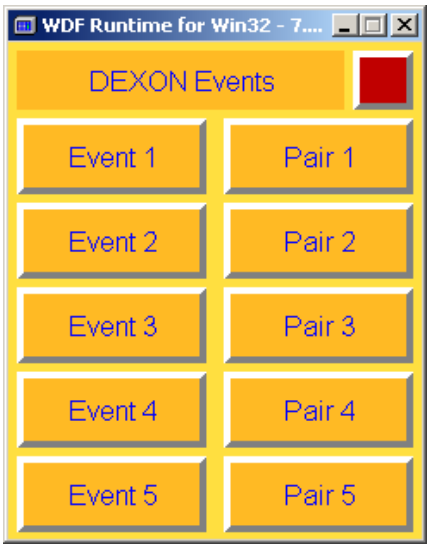
2.8. Cue モジュール

Cue モジュールは RS-232 ポートを介して、Cue Assistant または Assistant-S プロセッサと通信する機能を備えています。この通信は、シリアルモジュールによって実装されています。また、DXInterface プログラムの plugins ディレクトリにある Cue プログラムのサンプルを使用できます。

2.8.1.クイックスタート

モジュールの機能を確認するために、CD-ROM 内の Manuals/DXInterface/samples/cue にある Cue System Director ファイルである「DEXON_TEST.CSW」を開きます。このファイルは、ソフトウェア・バージョン 6.6.4.79 の Assistant-S デバイス用に作成されています。パネルファイル「TEST_PANEL.wdf」は、テスト用システムのボタンをシミュレートする用途で作成しています。コードは TEST_PANEL.cdw ファイルの中に書かれています。

1. 「Global Settings」タブにてサーバーの IP アドレスを設定します。
2. 「Serial」タブを選択します。
3. 「Enable Serial Communication」のチェックボックスを選択し、シリアルパラメータを入力します。使用する COM ポートを選択してチェックを入れます。今回の例では、ボーレート「115200」、データビット「8」、ストップビット「1」、パリティ「none」、ハンドシェイク「none」にしています。
4. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、そのパスを控えておきます。このファイル内のレイアウトは、有効なコマンドを受信すると表示されるようになります。
5. コマンドをキャッチするには、マッピングを作成する必要があります。マッピングでは、シナリオファイルと入力されるイベントコードの組み合わせでステップを定義します。マッピングを追加するには、Enabled を有効にし、入力イベントの名前を設定して、シナリオファイルを選択します。Event ボックスで必要なステップのシリアル番号または外部イベントコードを選択します。レイアウトが選択できない場合は、シナリオのパスが正しいことを確認してください。このサンプルで有効な入力イベントは、大文字小文字を区別する DXWALL_TEST.CSW が EVT1、EVT2、EVT3、EVT4、EVT5 となっています。
6. OK ボタンを押します。接続の確立に関するステータスが表示されます。タッチパネル上の右下にある小さなボタンが、接続が有効な場合は緑、接続を確立できなかった場合は赤になります。



イベントを送信すると、各マッピングによって定義されたレイアウトが表示されます。Cue システムは、イベントペア (Pair1, ...) を送信することができます。この場合、シナリオファイルのパスとステップは Cue コードによって直接定義されます。

2.8.2. プロトコル

Cue デバイスと DXInterface server 間のプロトコルメッセージはバイナリ形式を使用します。Cue プロトコルは、DXInterface のバイナリプロトコルの特別なサブセットです。クライアントは AMX または Creston などの他のクライアントと同じメッセージを送信できますが、サーバーは常に PONG、ACK、ERR メッセージのみで応答します。そのため、Cue デバイスは固定長のパケットのみを受け取ることになります。第 2 レベルのビデオ処理では、ウィンドウハンドルの代わりにユーザー定義のキーワードを使用する必要があります。

1. 各パーティはいつでも PING メッセージを送信できます。相手が利用可能であれば、PONG メッセージで応答する必要があります。
2. Cue デバイスが EVT メッセージを送信する際、EVT メッセージには可変長のイベントコードが含まれ、このコードは定義されたマッピングの Input パラメータと同じにする必要があります。
3. Cue デバイスが EVP メッセージを送信する際、EVP メッセージにはシナリオファイルとシナリオファイル内のステップを直接示す外部イベントコードまたはシリアル番号が含まれています。
4. EVT と EVP の両方で、システムは受信したことを通知する ACK メッセージを送信します。システムがリクエストを処理できなかった場合のエラーコードも含まれていることがあります。
5. システムが DISC 切断メッセージを送信することがあります。このメッセージは非同期のため、応答は必要ありません。DXInterface はこのメッセージで切断状態にあり、利用ができないことをクライアントに通知します。その後、クライアントは再接続を確認するため、PING メッセージを定期的に送信します。サーバー側が再び利用が可能になった場合には PING メッセージを送信します。
6. 任意のウィンドウの開閉を管理できます。

2.8.3. Cue サンプル

モジュールの機能を確認するために、CD-ROM 内の Manuals/DXInterface/samples/cue にある「DEXON_TEST.CSW」(Cue System Director ファイル) を開きます。このファイルは、ソフトウェア・バージョン 6.6.4.79 の Assistant-S デバイス用に作成されています。パネルファイル「TEST_PANEL.wdf」は、テスト用システムのボタンをシミュレートする用途で作成しています。コードは TEST_PANEL.cdw ファイルの中に書かれています。

このテストシステムには 10 個のボタンがあり、単純なイベント用のボタンが 4 個、イベントペア用のボタンが 4 個あります。2 つのボタンは、DXWall ライブラリの第 2 レベルの定義済みコマンドを使用して、ウィンドウの開閉が可能です。システムは起動時に接続パラメータを設定し、10 秒ごとに PING メッセージをサーバーに送信します。15 秒以内にサーバーからの応答がない場合、ステータスボタンは赤、それ以外は緑になります。

ボタンのサンプルコード

```
'Control Example for the DXWall(R) Controller
'
' ATTENTION: USE OF THIS SOFTWARE Is SUBJECT To THE FOLLOWING TERMS.
' The software is provided "AS IS" and without any warranty, expressed,
' implied or otherwise regarding its completeness or performance.

Module mAssistantS

'Runs at system startup
Sequence Autoexec

'Establish connection: Baud 115200, databits 8, no parity, 1 stop bit
DXWALL!DXinit(AssistantS.S5,"0192008n1",@ListenerOnline,@ListenerAck,@ListenerErr)
'This just shows last ACK or ERR message
AssistantS.A1.LevelOff()
AssistantS.A2.LevelOff()
'Set Device to Cue instead of Unknown
DXWall.SetModeDevice
'notify server about online status
DXWall.ping
'Update button
SequenceStart(DXWALL!Connected_Update)
'Start periodic ping
SequenceStart(@DXWALL!DXWatchdog)
End Sequence
Sequence ListenerOnline
'Update connected button
FastExe
If DXWALL!IsConnected() Then
ButtonValueSet(100,1)
Else
ButtonValueSet(100,0)
End If
EndFastExe
End Sequence
Sequence ListenerAck
'Last response was ok
FastExe
AssistantS.A1.LevelOn(:0.1)
AssistantS.A2.LevelOff(:0.1)
EndFastExe
End Sequence
Sequence ListenerErr
'Last response was error
FastExe
AssistantS.A1.LevelOff(:0.1)
AssistantS.A2.LevelOn(:0.1)
EndFastExe
```

```
End Sequence
Sequence TP01_Event1 Link 10001
FastExe
DXWall.SendEventEVT1
EndFastExe
End Sequence
Sequence TP01_Event2 Link 10002
FastExe
DXWall.SendEventEVT2
EndFastExe
End Sequence
Sequence TP01_Event3 Link 10003
FastExe
DXWall.SendEventEVT3
EndFastExe
End Sequence
Sequence TP01_Event4 Link 10004
FastExe
DXWall.SendEventEVT4
EndFastExe
End Sequence
Sequence TP01_OpenVideo Link 10005
FastExe
DXWall.OpenVIDEO1
DXWall.AssignVAR1
DXWall.END
EndFastExe
End Sequence
Sequence TP01_EventPair1 Link 10006
FastExe
DXWALL!DXSendEventPair("cuetest.dxs","STEP1")
EndFastExe
End Sequence
Sequence TP01_EventPair2 Link 10007
FastExe
DXWALL!DXSendEventPair("cuetest.dxs","STEP2")
EndFastExe
End Sequence
Sequence TP01_EventPair3 Link 10008
FastExe
DXWALL!DXSendEventPair("cuetest.dxs","STEP3")
EndFastExe
End Sequence
Sequence TP01_EventPair4 Link 10009
FastExe

DXWALL!DXSendEventPair("cuetest.dxs","STEP4")
EndFastExe
End Sequence
Sequence TP01_CloseVideo Link 10010
```

```
FastExe
DXWall.CloseVIDEO1
EndFastExe
End Sequence

End Module
**********************************************************************************************************
```

プロトコルの手順や機能の実装については、別のモジュールで説明します。

```
'Control Module for the DXWall(R) Controller '

' ATTENTION: USE OF THIS SOFTWARE IS SUBJECT To THE FOLLOWING TERMS.
' The software is provided "AS IS" and without any warranty, express,
' implied or otherwise regarding its completeness or performance.

Module DXWALL
Dim cmd As String * 400 ="¥EA¥E5¥00¥03EVT¥E5¥00¥03¥EB¥00"
'Saved channel address
Dim DXChannel As DeviceAddress
'This sequence is called to update the online status
Dim Connected_Update As Sequence
'Sent in case of error
Dim seqERR As Sequence
'Sent in case of error
Dim seqACK As Sequence

Dim Const_WatchdogPeriod As Integer = 5
Dim Const_WatchdogTimeout As Integer = 10

'Current watchdog phase
Dim Phase As Integer
'Current time
Dim Ticks As Integer
'Time when the last message arrived from the server
Dim TicksLast As Integer

'Initializes connection to the controller. It must be called before the
'use of any DX... functions or subroutines
'
'Parameters:
'Channel: Serial channel to use
'Str: Serial connection command
Public Sub DXInit(Channel As DeviceAddress, Str As String, Connected As Sequence,
seqACK0 As Sequence, seqERR0 As Sequence)
FastExe
DXChannel = Channel

'Store listeners
Connected_Update = Connected
```

```
seqACK=seqACK0
seqERR=seqERR0
'Setup connection
CommunicationSet(DXChannel, Str)
ReceiveEnable(DXChannel)
TriggerDisable(DXChannel)
CUEringDisable()
EventOnStringLengthSet (DXChannel,5, @DXReceived)
'Current phase of timer in seconds (last phase will send a ping message)
Phase=0
'Current ticks. We count current time to be able to check whether we are online or not
Ticks=1000
'Last tick when a message has arrived from the server.
TicksLast=960
EndFastExe
End Sub

'IsConnected function
'
'Returns: True if the connection seems to be alive.
Private Function IsConnected() As Boolean
IsConnected = Ticks-TicksLast<Const_WatchdogTimeout
End Function
'Disconnect procedure
'
'Sets the time of last value so, that the connection is considered to be
'disconnected
Private Sub Disconnect()
IfTicks>Const_WatchdogTimeout+1 Then
TicksLast = Ticks - Const_WatchdogTimeout
Else
Ticks = Ticks + Const_WatchdogTimeout
TicksLast = Ticks - Const_WatchdogTimeout
End If
End Sub
'Receive a 5 bytes packet
Private Sequence DXReceived
Dim RecStr As String * 5
Dim I As Byte
Dim checksum As Byte
FastExe
'moves 5 characters from receiving buffer To variable RecStr
ReceiveStringGet (DXChannel, RecStr, 5)
' StringSend(DXChannel,tmp)
checksum=0
For i=1 To 4
checksum = checksum + RecStr(i)
Next
If checksum = RecStr(5) Then
'checksum ok
```

```
If RecStr(3)=&HF1 Then
'DISCONNECT
Disconnect
' ReqSent = 0
SequenceStart(Connected_Update)
Else
If RecStr(3)=&HFF Then
'PONG
' ReqSent = ReqSent - 1
End If
If RecStr(3)=&H00 Then
'ACK OK
' ReqSent = ReqSent -1
'Put your acknowledge code here
SequenceStart(seqACK)
End If
If RecStr(3)=&H01 Then
'ACK ERR
' ReqSent = ReqSent -1
'Put your error handling code here
SequenceStart(seqERR)
End If
If RecStr(3)=&HFE Then
'PING
DXWall.pong
End If

TicksLast = Ticks
'Set status to connected
SequenceStart(Connected_Update)

End If
Else
DXWall.ERR
'checksum err
End If
EndFastExe
End Sequence
'Send a simple event protocol message to the controller
'
'Parameters:
'Scenario: The scenario file to launch
'Str: The event code to send to the server
Public Sub DXSendEventPair (Scenario As String, Str As String)
'Command length is maximum event length+1+6+3+2
'Dim cmd As String * 200 ="¥EA¥E5¥00¥03EVT¥E5¥00¥03¥EB¥00"
Dim chksum As Byte
Dim i As Integer
Dim pos As Integer
FastExe
```

```
'Set final length of the command
cmd(0)=(7+3+LEN(Str)+3+LEN(Scenario)+2) Mod &H100

'EVT
cmd(1) = &HEA 'EA
cmd(2) = &HE5 'E5 string
cmd(3) = &H00 'Length hi
cmd(4) = &H03 'Length lo
cmd(5) = &H45 'E
cmd(6) = &H56 'V
cmd(7) = &H50 'P
cmd(8) = &HE5 'E5 string
cmd(9) = &H00 'Length hi

'Set the variable length field of the EventCode field
cmd(10)=LEN(Str)

'Set length of string
For i=1 To LEN(Str)
cmd(i+10)=Str(i)
Next

'Set filename length
pos = 10+LEN(Str)+1
cmd(pos)= &HE5
pos = pos + 1
cmd(pos)= LEN(Scenario) / &H100
pos = pos + 1
cmd(pos)= LEN(Scenario) Mod &H100
pos = pos + 1

'Copy scenario filename
For i=1 To LEN(Scenario)
cmd(pos)=Scenario(i)
pos = pos + 1
Next

'Set command closing character
cmd(pos)=&HEB

'Count checksum
chksum = 0
For i=1 To pos
chksum = chksum + cmd(i)
Next
pos = pos + 1

'Set last byte to the checksum value
cmd(pos) = chksum
```

```
AssistantS.S5.StringSend( cmd)
'ReqSent = ReqSent + 1
EndFastExe
End Sub

'Send a simple ping message
'
'Parameters: none
'Returns: a value whether the DXWALL Controller responded in 1 second

Sequence DXWatchdog
Do While True
Wait(:1)
FastExe
Phase=Phase+1
Ticks=Ticks+1
IfTicks>32000 Then
'Use this to avoid overflows
If (Ticks -TicksLast)<1000 Then
Ticks = Ticks - TicksLast
TicksLast = 0
Else
Ticks=1000
TicksLast=960
End If
End If

If Ticks-TicksLast>=Const_WatchdogTimeout Then
'Connection lost
SequenceStart(Connected_Update)
End If


If Phase=Const_WatchdogPeriod Then
'Send a PING message Const_WatchdogPeriod seconds
Phase=0
DXWall.SendPing
'DXWall.ping
End If
EndFastExe
Loop
End Sequence
End Module
```

2.8.4. トラブルシューティング

接続できない
ハードウェアの接続を確認してください。インターフェースと Cue デバイスの両方が同じ通信設定を使用しているかどうかを確認してください。
Cue デバイスを再起動してください。サービスが実行されているか、モジュールが有効になっているかどうかを確認してください。
他のプログラムや DXInterface モジュールがシリアルポートを使っていないか確認してください。

DXInterface サービスに簡単なイベントを送信しているが、レイアウトが表示されない
configuration メニューの全てのタイプのログメッセージを有効にしてください。通信エラーのメッセージが表示されている場合は、Cue プログラムを確認してください。それ以外の場合は、マッピングが定義されていて有効になっているか、有効なデータを含んでいるかを確認してください。問題がなければ、configuration ウィンドウの「Global」タブの設定を確認してください。

DXWallServer の問題
本機との接続が問題ないか確認してください。「Global」タブで設定した IP アドレスと TCP ポートの値が正しいことを確認してください。

Binary または Text プロトコルの問題
詳細は 3. ウォールのプログラミング (WIL) (63 ページ) の章を確認してください。

2.9. SNMP モジュール

DXWall Interface アプリケーションは SNMP トラップを取り込む事ができます。SNMP トラップはエージェントからの SNMP メッセージを通してイベントの通知を可能にします。

2.9.1.概要

SNMPv1 トラップは RFC1157 で定義されており、以下のフィールドがあります。

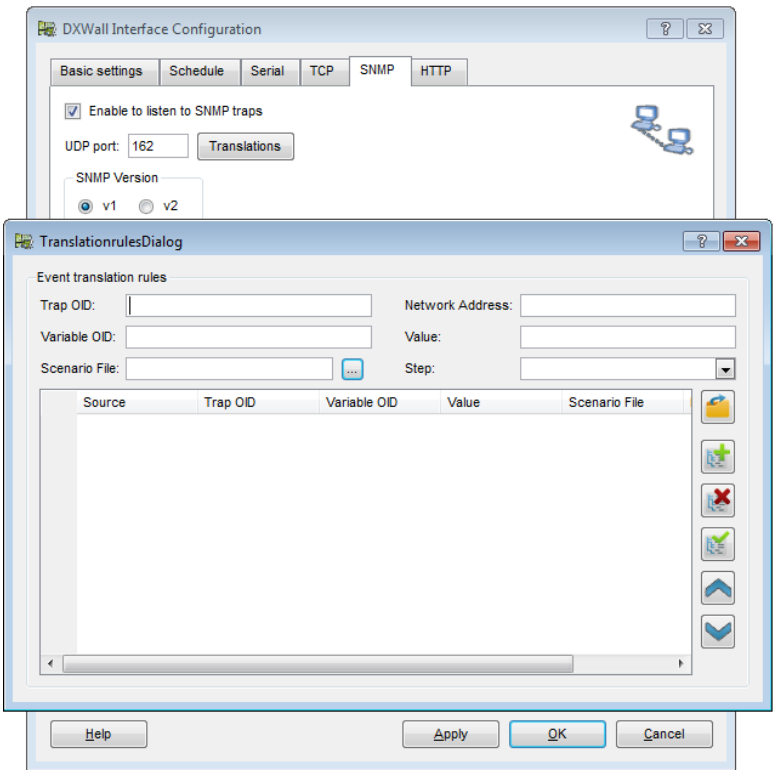
1. **Enterprise**
   トラップを生成する管理オブジェクトのタイプを識別します。
2. **Agent address**
   トラップを生成する管理オブジェクトの IP アドレスです。
3. **Generic trap type**
   Generic trap type フィールドの値を表示します。
4. **Specific trap code**
   特定のトラップコードを表示します。
5. **Time stamp**
   最後にネットワークを再初期化してからトラップが生成されるまでの経過時間です。
6. **Variable bindings**
   トラップを含む PDU のデータフィールドです。各 variable bindings は、特定の MIB オブジェクトインスタンスと現在の値を関連付けます。

2.9.2.クイックスタート

以下の手順にて、モジュールのテスト実行が可能です。

1. Configuration ダイアログの「SNMP」タブを選択し、「Enable SNMP connection」のチェックボックスにチェックを入れます。
2. ネットワーク機器のドキュメントを確認し、機器の仕様に基づいて SNMP ポートと SNMP のバージョンの値を設定します。SNMP ポートは通常、162 を使用します。他のシミュレーションソフトを使用することも可能です。
3. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、そのパスを控えておきます。このファイル内のレイアウトは、指定されたフィルタにマッチするトラップを受信した際に表示されるようになります。
4. 「Translations」ボタンを押します。トラップを取り込み、シナリオステップを起動するためのマッピングを作成します。マッピングでは、シナリオファイルと入力されるイベントコードの組み合わせでステップを定義します。マッピングを追加するには、入力アドレスを 192.168.1.100/0 の形式でデバイスの IP アドレスを設定します。その他のパラメータを <OPTIONAL> に設定します。シナリオファイルを選択し、イベントボックスに希望するステップのシリアル番号を選択します。レイアウトを選択できない場合は、シナリオパスが正しいか、保存されたシナリオにステップが含まれているか確認してください。
5. ネットワーク機器またはシミュレーションプログラムでトラップを生成します。

2.9.3. リファレンス



モジュールは SNMPv1 と v2 トラップを取り込むことができます。多くの SNMP イベントソースを定義できます。SNMP イベントソースは、1 つの SNMP トラップを定義し、それにイベントコードを関連付けます。SNMP イベントソースは以下のデータフィールドを使用します。

1. **Trap OID**
   標準的なトラップ (warmStart、linkDown、linkup、authenticationFailure、gpNeighbourLoss) 、または独自のトラップ OID です。ここに **<OPTIONAL>** を指定すると、このフィールドのテストが省略されます。
2. **Source address**
   トラップの送信元アドレスをフィルタリングできます。xxx.xxx.xxx.xxx/pppp (x が IP アドレス、pppp がソースポート) の標準的な UDP 形式を使用します。ソースポートが指定されていない場合は、そのポートに対するフィルタリングは行いません。ここに **<OPTIONAL>** を指定すると、このフィールドのテストを省略できます。
3. **Variable binding OID and Value**
   トラップの variable bindings の 1 つを表す OID の値です。ここに **<OPTIONAL>** を指定すると、このフィールドと Variable フィールドのテストが省略されます。Variable OID と Value の両方を指定すると、Value フィールドに格納されている値と一致する variable binding が存在する場合、トラップを受け入れます。Variable OID のみ有効で、Variable の値が **<OPTIONAL>** の場合、Variable binding OID が一致すると値に関係なくトラップを適用します。

トラップがフィルタに一致した場合、システムはそれを定義したマッピングエントリを見つけます。一致するエントリーが複数あった場合、システムは最初のマッピングを選択します。マッピングは以下を定義します。

1. 実行するシナリオファイル
2. シナリオファイル内ステップの外部イベントコードまたはシリアル番号 (#1)

マッピングが見つかると DXInterface は設定したステップでシナリオを起動します。

2.10. HTTP モジュール

HTTP モジュールでは、LAN で接続されているコンピュータのウェブブラウザから、定義済みのレイアウトを簡単に呼び出すことができます。

インストール CD-ROM の中に、DXInterface HTTP モジュールに関するサンプルビデオファイルがあります。

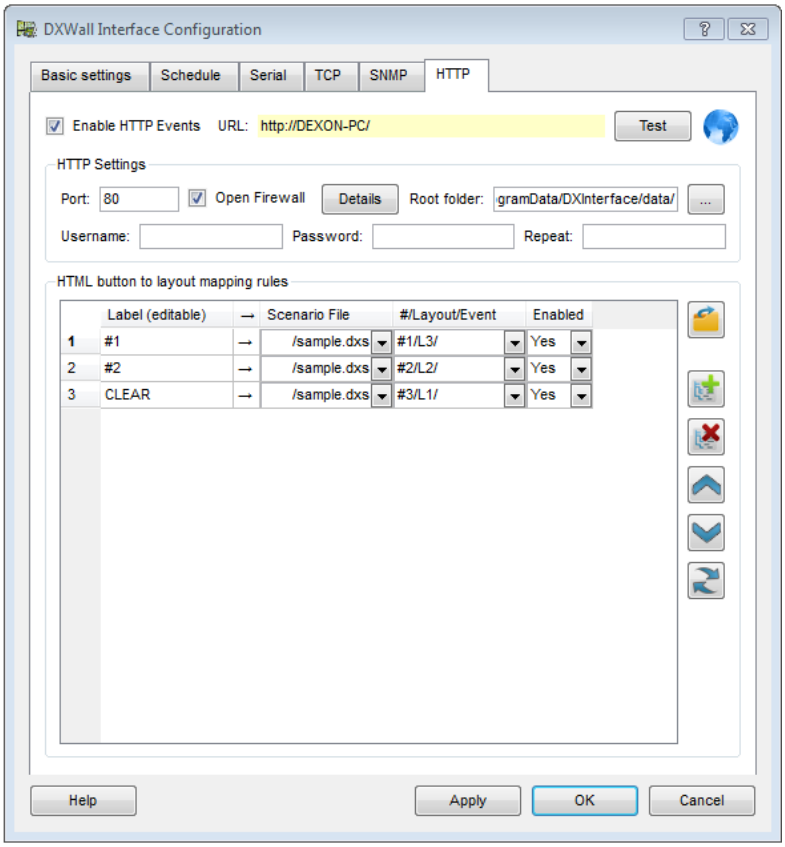
- Demos¥DXInterface_CreateScenario.wmv
- Demos¥DXInterface_HTTP.wmv
- Demos¥DXInterface_Firewall.wmv

2.10.1. クイックスタート

設定には DXWallControl アプリケーションの取り扱いに慣れている必要があります。設定するには、最初に configuration フォームの HTTP タブを開きます。

1. 「Enable HTTP events」のチェックボックスにチェックを入れます。使用するポート番号に「80」を入力します。
2. DXWallControl アプリケーションでシナリオファイルを作成します。ファイルを保存し、「Root folder」の欄に保存したフォルダのパスを記入します。
3. ユーザー名とパスワードは空欄のままにします。
4. 「Event definitions」パネルにて、シナリオファイルの相対パスを「File:」の欄に入力するか、または「…」ボタンでファイルを参照、選択してください。
5. 2 番目と 4 番目のステップに問題がない場合、「Event:」の欄に希望するステップのイベント名を入力します。「Label」欄には、ユーザーがそのステップを識別できるような空欄以外のラベルを入力できます。「Enabled」を有効にして、ボタンを押してください。
6. シナリオファイルから必要とされる全てのレイアウトに、上記の手順を繰り返し行います。
7. OK ボタンを押します。ウェブブラウザを開き、http://<server address>:80/ を入力してください。ページが表示され、各項目をクリックするとシナリオファイルのステップが実行されます。ローカルコンピュータは、常にサーバーアドレス「127.0.0.1」でアクセス可能です。

2.10.2. 詳細



Enable HTTP events
機能の有効 / 無効を設定できます。変更は「OK」ボタン、または「Apply」ボタンを押した後に適用されます。起動に失敗した場合は、システムログにエラーメッセージが記録します。起動後、http://<server name>/ の URL にアクセスできるようになります。デフォルトページには、定義済みのボタンが表示されます。http://<サーバー名>/<シナリオファイル名>.dxs?event=<イベント名> とブラウザに入力することで、定義済みのレイアウトを呼び出すことができます。

port
リモート接続を待ち受けるためにシステムが使用する HTTP のポート番号です。ほとんど場合、デフォルトの HTTP ポートは 80 になっています。このポートが他で使われている場合、8080、など他の有効なポート番号を設定できます。設定した値がその他のサービス、またはデーモンによって使用されていないことを確認してください。

URL
DXInterface によって作成されたページの URL を表示します。このアドレスをブラウザで使用します。ローカルコンピュータからアクセスする場合、ホスト名部分を IP アドレスや 127.0.0.1 に置き換えることが可能です。

Open Firewall / Details ボタン
Windows ファイアウォールのリモート接続を有効または無効に設定できます。また、リモートホストのページにアクセスできるように、ファイアウォールの設定を変更することも可能です。「詳細」ボタンを押すと、システムのファイアウォール設定を確認、変更することが可能です。

Root folder
この欄に作成済みのシナリオファイルを含むディレクトリのフルパスを入力します。

【注意】ここで指定したディレクトリ内の全てのファイルは、リモートユーザーからアクセス可能となりますのでご注意ください。ユーザーはこのディレクトリのファイルへのアクセスを no-access または read-only どちらかに設定可能です。ディレクトリはグローバルディレクトリの下に置くことを推奨します。

User name
パスワード確認の有効 / 無効を設定します。値が設定されていない場合、システムは全ての HTTP リクエストを受け入れて実行します。値が設定されている場合、システムは指定されたユーザー名での認証を求めます。ユーザー名に「: (コロン) 」の文字は使用しないでください。

Password / Repeat
上記の「User name」で指定したユーザー名のパスワードを設定します。パスワードのチェックをするために、2 箇所にパスワードを入力する必要があります。両方の欄に入力されたパスワードが一致したときにのみパスワードが設定されます。

Event definitions
HTTP サーバーは各シナリオステップを呼び出すボタンを含むページを表示します。これらのボタンはここで定義できます。ページの内容は下記の「Page Definition file」に記述されています。Event definitions グループの上部の欄には、現在の項目の値を表示します。

	リストに現在の値を追加します。テーブル内の項目をクリックすることで、フィールドにそのデータを読み込みます。
	ボタンを押すことで、選択中の項目を削除します。
	隣接する項目の並びを変更します。項目の順序は、モジュールが出力する HTML ページでもこの順序を使用するため重要になります。
	ボタンを押すことで、コンボボックス内のステップのリストを更新します。
	選択したシナリオファイルからステップをリストにロードします。

File
ルートディレクトリ内にあるシナリオファイルの相対パスを入力します。ここにパスを直接入力するか、または「Browse … 」からファイルを参照します。有効なファイルが指定されると、利用可能なイベント名の値がイベントコンボボックスに表示されます。ファイル名は、先頭の「/ (スラッシュ) 」の有無に関わらず有効です。

 # / Step / Event
シナリオファイル内のレイアウトを識別します。シリアル番号や外部イベントコードを使用します。コンボボックスからレイアウトを選択します。見つからない場合は、シナリオファイルのパスが正しいことを確認してください。

Label
項目ごとにラベルを設定できます。ページ内のボタンのラベルやリンクテキストに使用できます。

Enabled
項目の有効 / 無効を設定します。無効の場合、設定内容は保存されますが、出力されるウェブページには表示されません。

Page definition file

Page definition file (<root folder>/pagedef.html) は、イベントのコンテナページがどのように見えるかを記述したものです。各イベントは HTML ボタンまたはリンクにマッピングされます。インストール時にはデフォルトファイルが作成されますが、必要に応じて変更することが可能です。フォーマットはシンプルな HTML にいくつかのキーワードを加えています。ファイルには、静的な項目 (ヘッダーやフッターなど) と、ボタン固有の項目があります。各ボタンには HTML コードがあり、1 つずつ出力にコピーされます。ボタンはグループ化することもできます。ボタンをグループ化すると、HTML のテーブルや段落などに配置する際に便利です。キーワードは以下の通りです。

- **$DXROWLEN$**
  ボタンのグループサイズを指定します。サイズは$DXROWLEN$の後の 10 進数で表記します。キーワードは定義ファイルの最初の行に指定する必要があります。サイズが指定されていない場合、デフォルトの値は「2」になります。
- **$DXTRSTART$**
  キーワードより前の HTML コンテンツは、出力 HTML に一括してコピーされます。このキーワードは必須です。
- **$DXTD$**
  $DXTRSTART$ の終わりと $DXTD$ の始まりとの間にあるコンテンツは、すべてのボタングループの前に出力へコピーされます。HTML コードは通常、テーブルのボタンの各行の前に <tr> タグを定義するために使用されます。このキーワードは必須です。
- **$DXTREND$**
  $DXTD$と$DXTREND$との間のコンテンツには、各ボタンのコードが含まれています。このコードは通常、イベントのプロパティを持つリンクまたはフォームを含める必要があります。定義済みのレイアウトを呼び出すには、http://<サーバー名>/<シナリオファイル名>.dxs?event=<イベント名> を呼び出す必要があります。通常、サーバー名、シナリオファイル、イベントのパラメータはイベント定義ファイルで指定します。
- **$DXADDRESS$**
  ブラウザ側の http サービスのアドレスです。これを使用して、有効で環境に依存しない IP アドレスの値を確立します。このキーワードは任意ですが、機能性の面から使用が推奨されます。
- **$DXSCENARIO$**
  現在のイベントのシナリオファイルの相対パスです。このキーワードは任意ですが、機能性の面から使用が推奨されます。
- **$DXLABEL$**
  イベントのラベルです。このキーワードは任意ですが、機能性の面から使用が推奨されます。
  $DXADDRESS$、$DXSCENARIO$、$DXLABEL$、$DXEVENT$ は、現在のシステムアドレス、シナリオファイル、ラベル、イベントの値に置き換えられます。イベント定義リストに記載されていない組み合わせを指定することは、それらが参照するシナリオファイルがルートディレクトリに配置されている場合には許可されますが、モジュールはそれらのキーワードを置き換えません。それらは静的コンテンツ (ヘッダーまたはフッター) のいずれかに配置する必要があります。
- **$DXEVENT$**
  現在のイベントのイベント名です。これは「Event」の欄で指定したものと同じ文字列になります。このキーワードは任意ですが、機能性の面から使用が必要とされています。
- **$DXEND$**
  $DXTREND$ と $DXEND$ との間のコンテンツには、グループの終了タグが含まれます。
  通常、グループの終了タグとして </tr> タグを記述することができます。$DXEND$ の後のコンテンツは、1 つのブロックで出力にコピーされます。このキーワードは必須です。
- **$FORMID$**
  フォームまたはステップの順序の番号です。

サービスは次のように動作します。
モジュールが pagedef.html を開きます。最初に$DXTRSTART$キーワードの前のヘッダーをコピーします。その後それを繰り返し、各イベント定義のボタンコンテンツをコピーして、キーワード $DXADDRESS$、$DXSCENARIO$、$DXLABEL$、$DXEVENT$ を現在の値に置き換えます。ボタングループの境界が認識された場合、前のグループの終了コンテンツと新しいグループの開始コンテンツが出力にコピーされます。最初のボタンの前にはグループ開始のコンテンツのみが書き込まれ、最後のボタンの後にはグループ終了のコンテンツが書き込まれます。

Page Definition file のサンプル：
デフォルトのインストールディレクトリ内の data の中に、編集可能なファイルがあります。

```
$DXROWLEN$ 2
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
	<title>HTTP Control</title>
	<style type="text/css">
		<!--
	* {margin:0; padding:0;}
	body {background-color:#d4d4d4; font-family: Tahoma, Verdana; color: #6d6458; font-size:
0.8em;}
	table {margin:0 auto;}
	table tr{height:100px;}
	table td{vertical-align:middle;text-align:center;}
	#slogan {font-size:1.2em;width:450px;line-height:75px;color:#fff;}
	#logo {position:absolute;left:580px;top:0px;overflow:hidden;width:200px;height:70px;}
	#container {float:left;width:780px;position: absolute;left: 50%;margin-left: -390px;margin-
top:30px;border:3px solid #000;}
	#header {overflow:hidden;background-
image:url(header.png);width:760px;height:75px;padding-left:20px;}
	#content {background-color:#8c8e8c; width:780px;min-height:182px;padding-top:40px;}
		-->
	</style>
</head>

<script type="text/javascript">
...
</script>
<body>
	<div id="container">
		<div id="header">
			<div id="slogan">
				Company slogan</div>
			<div id="logo">
				<img width="200px" src="/logo.png" align="center" alt="Company Logo"
/></div>
```

```
            </div>
            <div id="content">
                <center>
                    <table width="70%">
                        $DXTRSTART$
                        <tr>
                            $DXTD$
                            <td>
                                <form method="get" action="$DXADDRESS$/$DXSCENARIO$"
name="form_$FORMID$" id="Form1">
                                <div style="margin: 0 auto; position: relative; width: 160px;
height: 75px;">
                                    <div
onclick="javascript:document.form_$FORMID$.submit();" style="cursor: pointer;
                                        position: absolute; padding-top: 30px; width: 160px;
text-align: center; font-weight: bold;
                                        color: #ffffff;">
                                        $DXLABEL$
                                    </div>
                                    <a href="javascript:;"
onclick="javascript:document.form_$FORMID$.submit();">
                                        <img id="$DXEVENT$" border="0" src="button_off.gif"
alt="" />
                                    </a>
                                    <input type="hidden" name="event" value="$DXEVENT$"
id="Hidden1" />
                                </div>
                                </form>
                            </td>
                            $DXTREND$
                        </tr>
                        $DXEND$
                    </table>
                </center>
            </div>
        </div>
</body>
</html
```

HTML テーブルにボタンを配置したページを作成します。テーブルの幅は 2 ($DXROWLEN$) で、各ボタングループは <tr> タグを使用してテーブルの 1 行を表示します。ボタンは入力フォームとして表示され、イベントパラメータは hidden 要素が指定されています。

2.10.3. トラブルシューティング

ASCII でない文字セット
DXInterface は UTF-8 の文字セットをサポートしています。使用する言語でラベルを自由に設定できます。Internet Explorer が文字を正しく表示できない場合は、Internet Explorer の「表示/エンコード」メニューの「Unicode (UTF-8) 」がチェックされていることを確認してください。

モジュールを有効または再起動する前に、モジュールに接続しているすべてのクライアント (ブラウザ) を閉じてください。ブラウザによっては、モジュールがポートからの接続を解除できない場合があります。

Windows ファイアウォール
モジュールを有効にする前に、指定されたポートを使用している他のプログラムがないことを確認してください。また、ファイアウォール機能によってポートの使用が制限されている場合もあります。「OK」または「適用」ボタンを押すと、警告メッセージが表示される場合があります。Windows のファイアウォールの設定にて、サービスのアクセスを許可するように設定してください。

セキュリティについては、「Root folder」と「User name」の説明を参照してください。

ヒント：
DXWallControl のステップのイベント名などは判別しやすいものを設定してください。後で簡単に選択できることで、設定ミスを防止できます。

3. ウォールのプログラミング (WIL)

3.1. クイックスタート

3.1.1.WIL のイントロダクション

DXInterface のプロトコルは WIL (Windows interface language) と呼ばれ、全ての入力デバイスに対して使用可能です。WIL はコマンド属性と構造を定義した高レベルのコマンドセットで、TCP またはシリアル通信で使用できます。低レベルのパケットのフォームは WIL とは独立しています。これにはテキストまたはバイナリのパケットが使用されます。WIL に共通する特徴は以下になります。

- WIL の目的はウォール上のビデオレイアウトを変更するコマンドセットを定義することです。
- 実行は常にシーケンシャルに処理されます。
- ウィンドウハンドルコマンドは同期的に送信されます。パラメータのクエリーコマンドは非同期です。
- 明るさの値やウィンドウハンドルなどの情報を変数に格納できます。
- WIL の一連のコマンドはセッションとして定義されます。各セッションはプライベートな変数を持ちます。
- セッションは TCP 接続、シリアル接続で使用できます。
- 各セッションはプロパティの変更コマンドの処理をグループ化できます。これらの処理はコマンドグループの終わりを意味する END コマンドを送ったときに同時に処理されます。
- セッションでは $GLOBAL.name (name 部分はユーザー定義の名前です) のようなグローバル変数も使用できます。
- セッションはウィンドウリストのコピーを保存します。UPD または GET WINDOWS コマンドでこのリストの問い合わせや変更ができます。

WIL コマンドセットは以下のモジュールで使用できます。

- TCP モジュール
- SNMP モジュール
- 特殊機器用 シリアルモジュール (AMX、Crestron、Cue)
- HTTP モジュール

TCP の場合、最初に TCP 通信を有効にする必要があります。詳細については、2.5.1 章の「概要とクイックスタート」を参照してください。インターフェースが設定されていれば、任意のターミナルプログラムや独自のアプリケーションを使って、TCP ポートを介して接続を開くことができます。本機には簡単な TCP ターミナルプログラムが用意されており、以下の形式で実行できます。

dxlink.exe <TCP / IP アドレス> -p <TCP ポート>.

3.1.2.スクリーンレイアウトの選択

シナリオステップと呼ばれる事前に設定されたレイアウトから選択可能です。これらのステップは DXWallControl アプリケーションを使って作成できます。「TCP」タブの「Enable TCP Communication」を有効にすることで、DXInterface からシナリオステップを起動できます。例えば、入力コマンド INP1 は C:¥scenarios¥scenario.dxs のシナリオステップ「Step 1」を常に起動するように定義したマッピングを含んでいるとした場合、そのステップを起動させるには以下のコマンドを実行します。

EVT INP1

この件に関しては 2.5.1 概要とクイックスタート (26 ページ)をご確認ください。

3.1.3.手動でのウィンドウのオープンと管理

最もよく使われるコマンドは OPN、CLO、SEL です。これらのコマンドはビデオウィンドウを開いたり、閉じたりするときに使用します。SEL WND コマンドでは、既存のウィンドウのサイズ、明るさなどのパラメータを変更できます。SEL SRC コマンドでは、デフォルトソースのパラメータを取得することができます。詳細については、4.2 「テキストプロトコルを使った手動でのウィンドウの開閉」を参照してください。3.5 アプリケーション層 – WIL コントロールメッセージ の章では、各コマンドの説明と使用例が記載されています。

```
OPN SRC1
SET RESULT $GLOBAL.MYVIDEO1
SET WIDTH 800
SET HEIGHT 600
END

SEL WND $GLOBAL.MYVIDEO1
SET WIDTH 1024
SET HEIGHT 768
END

CLO $GLOBAL.MYVIDEO1
```

3.1.4.スクリプトからのウィンドウのオープンと管理

4.1 コマンドラインからのウィンドウのオープン、クローズの方法 (100 ページ) では、スクリプトを使用してビデオウィンドウを開閉する例を紹介しています。

3.2. イントロダクション

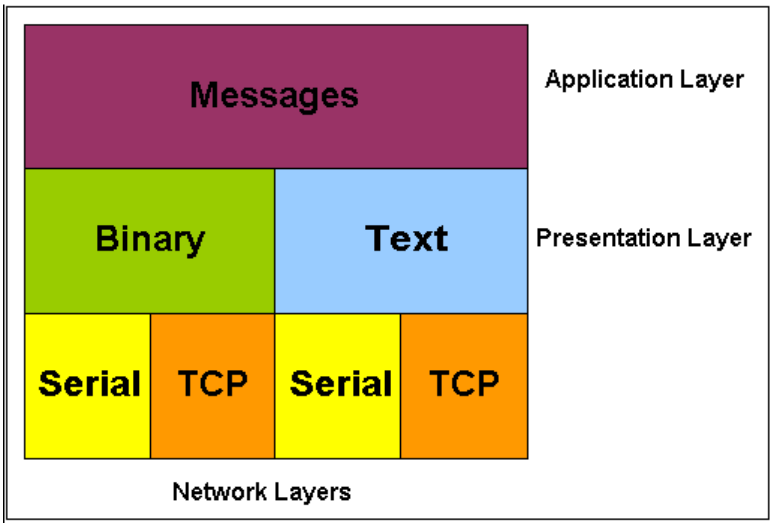
DXInterface のプロトコルは Window Interface Language (WIL) と呼ばれており、全ての入力デバイスに対して使用できます。WIL は静的なスクリプトもサポートしています。
WIL は、ウォールの第 2 レベルのプログラミングの一部です。アプリケーション層はメッセージベースで、全てのバリエーションにおいてほとんど同じです。クライアントがリクエストを送信し、サーバーがこれらのリクエストに応答します。
いくつかの特別なケースでは、サーバーがメッセージを送信する場合があります。全てのメッセージは、最初の値が常にメッセージのタイプを表し、その他の全ての値が任意のパラメータになります。WIL メッセージはバイナリパケットにパッキングされるか、テキストで表現されます。
例えば、イベントコード「EVT220」を送信するメッセージは、バイナリ・プレゼンテーションレイヤーを使用した場合、以下のようになります。非表示文字は 16 進コードで表しています。

EA E5 00 03 'E' 'V' 'T' E5 00 06 'E' 'V' 'T' '2' '2' '0' EB CS

テキストプロトコルが使用する場合は、以下のようになります。

EVT EVT220**¥r¥n**

¥r¥n は改行文字 CR / LF を ASCII コードの 13 と 10 に置き換えたものです。アプリケーション層のメッセージは常に同じものですが、それぞれのフォーマットで表されています。また、データリンクのタイプも自由に選択できます。
テキストプロトコルとバイナリプロトコルの両方を、TCP またはシリアルで送信可能です。初期設定では、テキストプロトコルが設定されています。これは、多くの環境での動作実績があり、プログラムや dxlink などのターミナルソフトウェアを使った手動テストを簡単に行えるため、テキストプロトコル設定のままで使用することをお勧めします。
クライアントがより多くのリソースを持ち、短い開発期間で開発する場合は、テキストプロトコルのほうがよいです。ターミナルプログラムにより手動でテストできます。



3.3. プレゼンテーション層 - バイナリプロトコル

バイナリプロトコルはフィールドをパケットにまとめます。プロトコルメッセージは以下の書式になります。全ての値は 16 進数です。

EA	**E5**	Hi	Lo	Str…	**E1**	…	**E3**	…	**EB**	CS

1. メッセージは常にプロトコルバイト **EA** で開始します。
2. プロトコルメッセージの各フィールドは、タイプ識別子で始まります。最初のフィールドには、常にメッセージタイプの値が含まれています。全てのフィールドタイプを以下に示します。
3. E5：Latin-1 文字列フィールドです。文字列の長さは E5 バイトの後、データバイトの前の 2 バイトの unsigned short 型の値で表します。以下の例は 3 バイトの文字列「ING」を表しています。

E5	00	03	00	69	5E	67

4. 文字列フィールドは、後述の「:String」によって定義されます。
5. E6：Unicode 文字列フィールドを表します。このフィールドタイプも文字列を表しますが、UCS2 エンコーディングとビッグエンディアンのバイトオーダーです。文字列の長さは E5 バイトの後、データバイトの前の 2 バイトの unsigned short 型の値で表します。以下の例は 3 バイトの文字列「UCS」を表しています。

E6	00	03	00	55	00	43	00	53

6. Unicode 文字列フィールドは、後述の「:String」によって定義されます。クライアントはどのタイプの文字列を使用するかを指定します。
7. E1：unsigned byte 値です。10 進法の 10 は以下のように表されます。

E1	0A

8. このようなフィールドは、後述の「:Byte」で定義されます。
9. E2：unsigned double byte 型の値です。MSB が最初のバイトに入ります。例えば、16 進数「0x0203」は以下のようになります。

E2	02	03

10. このようなフィールドは後述の「:Short」で定義されます。
11. E3：unsigned の 4 バイトの値です。MSB が最初のバイトに入ります。例えば、16 進数「0x30010305」は以下のようになります。

E3	30	01	03	05

12. このようなフィールドは後述の「:int」で定義されます。
13. E4：unsigned の 8 バイトの値です。MSB が最初のバイトに入ります。例えば、16 進数「0x7060504030200103」は以下のようになります。

E4	70	60	50	40	30	20	01	03

14. このようなフィールドは後述の 「:Long」で定義されます。
15. E7：PNG のデータフィールドです。このフィールドタイプは、 PNG 画像データを表します。データの長さは、E7 バイトの後、データバイトの前の 2 バイト unsigned short 型の値で表します。長さフィールドはバイト単位で表されます。以下の例は、PNG 画像の先頭部分を示しています。

E7	00	BA	89	50	4e	47	0D	0A

16. メッセージは **EB** コードと**チェックサム**値で終了します。チェックサムは、EA と EB のセパレータの間のバイトと、それを含む 8 ビットの合計です。チェックサムの値が 0 であるのは特別です。これは、パケットがチェックサムの値を持っていなく、このバイトは省略されることを意味しています。これはテスト目的で実行されます。DXInterface サービスは、常に実際のチェックサム値を持つパケットを送信します。

メッセージの最初のフィールドは常に文字列、またはメッセージのタイプを表すバイトフィールドです。正しい値については「アプリケーション層 – プロトコルメッセージ」の章に記載されています。ほとんどのメッセージはテキストモードと共通です。
SET ENCODING、SET MODED、TST メッセージだけはバイナリモードのみで使用可能です。

サーバーは常に **Latin-1** と **Unicode** の両方の文字列を受け入れます。クライアントは UCS2 文字列を受け入れるかどうかを設定可能です。「SET ENCODING ASCII」コマンドが送信された場合は、Latin-1 文字列だけがクライアントに送信されます。クライアントが「SET ENCODING UCS2」を設定した場合、クライアントは E5 と E6 の両方の文字列のタイプを受信する用意が必要です。
メッセージタイプの文字列は、常に E5 タイプの文字列です。

「SET ENCODING UCS2」とチェックサムの 16 進コードは以下になります。

¥ea¥e5¥00¥03¥53¥45¥54¥e5¥00¥08¥45¥4e¥43¥4f¥44¥49¥4e¥47¥e5¥00¥04¥55¥43¥53¥32¥eb¥e3

「SET ENCODING ASCII」とチェックサムの 16 進コードは以下になります。

¥ea¥e5¥00¥03¥53¥45¥54¥e5¥00¥08¥45¥4e¥43¥4f¥44¥49¥4e¥47¥e5¥00¥05¥41¥53¥43¥49¥49¥eb¥30

Cue デバイスは特別なバイナリプロトコルを使用します。詳細については Cue モジュールの章を参照してください。「SET MODED」メッセージで、通常のプロトコルと Cue 用プロトコルを選択できます。Cue モジュールはこのプロトコルをデフォルトで使用します。

「SET MODED CUE」とチェックサムの 16 進コードは以下になります。

¥ea¥e5¥00¥03¥53¥45¥54¥e5¥00¥05¥4d¥4f¥44¥45¥44¥e5¥00¥03¥43¥55¥45¥eb¥c1

「SET MODED UNKNOWN」とチェックサムの 16 進コードは以下になります。

¥ea¥e5¥00¥03¥53¥45¥54¥e5¥00¥05¥4d¥4f¥44¥45¥44¥e5¥00¥07¥55¥4e¥4b¥4e¥4f¥57¥4e¥eb¥18

「TST」メッセージはテスト用です。16 進コードは ¥ea¥e5¥00¥03¥54¥53¥54¥eb¥b8 です。
サポートされている全てのタイプのフィールド値を含んだテストメッセージを返信します。

3.4. プレゼンテーション層 - テキストプロトコル

テキストプロトコルは、同じアプリケーション層のメッセージをエンコードします。各メッセージはメッセージの種類を表す 3 文字のメッセージ識別子で開始します。識別子については、「アプリケーション層 - WIL コントロールメッセージ」の章に記載されています。メッセージは改行 (¥r¥n ASCII 13 と 10) で区切られ、UTF-8 形式でエンコードされます。UTF-8 形式では、標準文字 (ASCII コード 0-127) はそのままの形で、それ以外の文字はより多くのバイトで対応します。

メッセージタイプと改行の間にあるその他のパラメータは、すべてスペースで区切られます。これらのパラメータのフォーマットはメッセージタイプによって文字列、16 進数、または 10 進数のいずれかになります。スペースを含むパラメータを指定する場合は、ダブルクウォート (“) で囲む必要があります。このような文字列では、「¥”」のようにエスケープしてダブルクウォートを付けることができます。
以下が例です。

SET FONT Arial
[SET] [FONT] [ARIAL] の 3 つのブロックとして解釈されます。

SET FONT “MS Mincho”
[SET] [FONT] [MS Mincho] の 3 つのブロックとして解釈されます。

SET OSD[0].TEXT “I am a ¥“Controller¥””
[SET] [OSD[0].TEXT] [I am a Controller] として解釈されます。

整数値は、10 進数または 16 進数で表されます。16 進数は「0x」で始まる必要があります。また、サーバーはこれらの値を 2 つの形式で返すことができます。0x で始まる値は常に 16 進数で、それ以外は 10 進数です。

putty などのサードパーティ製のターミナルソフトでテストできます。その際は Raw モードで使用する必要があります。これらサードパーティ製ツールの使用は、弊社では責任を負いかねますのでご了承ください。本機には「dxlink」というターミナルアプリケーションを用意しています。詳細については、「TCP モジュール」の章を参照してください。

テキストプロトコルは、バイナリプロトコルにはない 2 つの特別なコマンドを持っています。
「**QUIT**」コマンドは、サーバーに TCP ソケットを閉じるように指示します。シリアル接続ではこのコマンドは使用しないでください。
「**HELP**」コマンドは、一般的なヘルプテキストを表示します。「HELP <cmd>」は、ヘルプがあれば指定されたコマンドのヘルプを表示します。

EVT、SET、BRIGHTNESS のような予約されたメッセージコードは、大文字と小文字を区別しません。

3.5. アプリケーション層 – WIL コントロールメッセージ

この章ではメッセージタイプの説明をします。特に表記がない場合、通常これらはバイナリとテキスト両方のモードで使用可能です

メッセージには 2 つのタイプがあり、いくつかはアトミック (処理が完了するまで他の処理が途中で割り込むことができない操作) であり、メッセージグループの一部であるものもあります。メッセージグループは接頭コード、接尾コード、本体メッセージで構成されています。

Clear wall (CLR) – アトミックメッセージ

ウォール全体をクリアします。	
詳細	このメッセージの開始と完了との間には遅延があります。 このメッセージとサーバー接続の間には排他制御があり、クリアコマンドの実行が開始されたときに、開いている既存のウィンドウすべてを閉じます。
形式	(コマンド:String)
コマンド	CLR
返り値	サーバーは ACK (acknowledge メッセージを返します。
サンプル	CLR¥r¥n **¥ea¥e5**¥00¥03¥43¥4c¥52**¥eb**¥9e

シンプルイベント (EVT) – アトミックメッセージ

イベント文字列をサーバーに送信します。	
詳細	これは通常、マッピング構造のインデックスです。 サーバーはイベントコードをインデックスとして使用し、事前に定義されたレイアウトを表示します。 詳細については「マッピング」の説明を参照してください
形式	(コマンド:String, イベントコード:String)
コマンド	EVT
返り値	サーバーは ACK (acknowledge メッセージを返します。
サンプル	EVT EVT1¥r¥n **¥ea¥e5**¥00¥03¥45¥56¥54**¥e5**¥00¥04¥45¥56¥54¥31**¥eb**¥b5

イベントペア (EVP) – アトミックメッセージ

シナリオファイルのパスとステップ識別子 (外部イベントまたはシリアル番号) の組み合わせをサーバーに送信します。	
詳細	サーバーは DXWallControl で外部イベントコードが設定されたシナリオファイルのレイアウトを表示します。
形式	(コマンド:String, シナリオパス:String, 外部イベント:String)
コマンド	EVP
シナリオパス	シナリオファイルのパス。最大長 255 バイト。
外部イベント	シナリオファイル内の外部イベントコード､またはステップのシリアル番号を 1 つ指定。最大長 100 バイト。
返り値	サーバーは ACK (acknowledge メッセージを返します。
サンプル	EVP #1 scenario.dxs¥r¥n C:ProgramData¥DXInterfacedata¥scenario.dxs の最初のステップを実行します。

ブロック・イベントペア (EVB) – アトミックメッセージ

シナリオファイルのパスとステップ識別子 (外部イベントまたはシリアル番号) の組み合わせをサーバーに送信します。	
詳細	サーバーは DXWallControl で外部イベントコードが設定されたシナリオファイルのレイアウトを表示します。システムが起動している場合は、DXInterface がコマンドを保存し、最後の EVB コマンドを実行します。
形式	(コマンド:String, 外部イベント:String, シナリオパス:String)
コマンド	EVB
シナリオパス	シナリオファイルのパス。最大長 255 バイト。
外部イベント	シナリオファイル内の外部イベントコード、またはステップのシリアル番号を 1 つ指定。最大長 100 バイト。
返り値	サーバーは ACK (acknowledge) メッセージを返します。
サンプル	EVB #1 scenario.dxs¥r¥n C:ProgramData¥DXInterfacedata¥scenario.dxs の最初のステップを実行します。

Acknowledge (ACK, ERR) – アトミックメッセージ

クライアントの要求に対する応答として、サーバーがこのメッセージを返信します。		
詳細	Cue デバイスとその他の機器では相違があります。 Cue モジュールは特別なバイトコード化したメッセージを文字列コードの代わりに使用します。詳細については「プレゼンテーション層 – バイナリプロトコル」を参照してください。 エラーコードが返された場合、可読メッセージがエラーとしてログに記録されます。	
形式	(コード:String<, エラーコード:int>) または (コード:byte)	
サンプル	OK : 要求が問題なく実行された。**(ACK)**	通常 : ACK¥r¥n **¥ea¥e5**¥00¥03¥41¥43¥4b**¥eb**¥8c Cue : **¥ea¥e1**¥00**¥eb**¥b6
	エラーが発生 **(ERR)**	通常 : ERR 80000001¥r¥n **¥ea¥e5**¥00¥03¥41¥43¥4b**¥e3**¥80¥00¥00¥01¥eb¥00 Cue : **¥ea¥e1**¥01**¥eb**¥b7
	チェックサムエラー : コマンドは問題なく解析されたが、チェックサム値が有効ではない。 **(ERR CHKSUM)**	Cue : **¥ea¥e1**¥02**¥eb**¥b8

Ping (PING) – アトミックメッセージ

応答確認用の Ping メッセージを送信します。	
詳細	PING メッセージはサーバーとクライアントの両方から送信可能です。相手は PONG メッセージによって 1 秒以内に応答します。応答がない場合は接続が切断されている可能性があります。
形式	(コード:String) または (コード:byte)
コード	0FEh
サンプル	PING¥r¥n **¥ea¥e5**¥00¥04¥50¥49¥4e¥47**¥eb**¥ec
Cue サンプル	**¥ea¥e1**¥fe**¥eb**¥b4

Pong (PONG) – アトミックメッセージ

PING メッセージに対する応答メッセージです。	
詳細	PING メッセージはサーバーとクライアントの両方から送信可能です。相手は PONG メッセージによって 1 秒以内に応答します。応答がない場合は接続が切断されている可能性があります。
形式	(コード:String) または (コード:byte)
コード	0FFh
サンプル	PONG¥r¥n **¥ea¥e5**¥00¥04¥50¥4f¥4e¥47**¥eb**¥f2
Cue サンプル	**¥ea¥e1**¥ff**¥eb**¥b4

Open window (OPN) – オープンメッセージグループ

ウォール上にウィンドウを開きます。	
詳細	このメッセージはアトミックではありません。このメッセージはウィンドウを開く手順を開始します。手順完了時にウィンドウに表示する入力ソースを選択します。 入力ソースは ID、名前、またはインデックスによって指定可能です。「GET SOURCES」コマンドでソースをリスト表示できます。 「GET SOURCES」はソースの ID と名前の組み合わせのリストを返します。入力ソースをインデックスで選択する場合、クライアントは直近の「GET SOURCES」コマンドの結果リストから、希望するソースの行インデックスを指定する必要があります。インデックスは 0 から始まります。
サンプル	GET SOURCES BEG SRC 0x00000000 VIDEO1 SRC 0x02000000 VIDEO2 SRC 0x01000000 VIDEO3 SRC 0x01000001 VIDEO4 SRC 0x04000000 VIDEO5 SRC 0x04000001 VIDEO6 SRC 0x05000000 APPLICATION END 3 行目に表示されるビデオソースは ID 指定では「**OPN** 0x01000000」、インデックス指定では「**OPN $S[2]**」、名前指定では「**OPN VIDEO3**」によって選択できます。 ソースの選択に成功した場合、DXInterface は何も応答しません。クライアントは、サイズなどのデフォルトのパラメータを変更できます。詳細は SET メッセージを参照してください。 open コマンドは「END」メッセージでコミットされ実行されます。エラーがあった場合、サーバーは ID またはエラーコードを表示します。エラーコードはログの中に説明が記録されます。 OPN 0x01000000 SET WIDTH 800 SET HEIGHT 600 END HEX 0x0b000001

返されたウィンドウハンドルを使って、後でウィンドウを参照することができます。クライアントによってはハンドルを保持できない場合があります。これを解決するために 3 つの方法があります。

$A キーワード

$A は最後に開いたウィンドウハンドルを表します。

```
OPN 0x01000000
SET WIDTH 800
SET HEIGHT 600
END
HEX 0x0b000001
CLO $A
ACK
```

$ キーワード

$S、$A、$W といった予約されたキーワードを除く、「$」で始まる任意のキーワードを使ってウィンドウをハンドルする方法です。ウィンドウを開く際に、SET RESULT <$keyword name> コマンドを指定します。これらのキーワードはプライベートです。各セッションはこれらのキーワードを保持する領域を持っています。セッションとは、TCP ソケット接続または peer-to-peer のシリアル接続を意味します。

$GLOBAL.で始まるキーワード変数は、全てのセッションで表示されるグローバルキーワードです。メモリリークを防ぐため、これら RMV<$GLOBAL.name>コマンドを使って手動で削除する必要があります。変数名は大文字と小文字を区別しません。DXWall server の設定が変更されると、全ての定義済みキーワードが削除されることに注意してください。

```
OPN 0x01000000
SET WIDTH 800
SET HEIGHT 600
SET RESULT $VIDEO1
END
HEX 0x0b000001
CLO $VIDEO1
ACK
```

GET WINDOWS コマンド

現在のウィンドウリストを取得する方法です。返された値を保存することで、クライアント側でリストを作成し、任意のウィンドウを閉じることが可能です。リストのインデックスキーワード $W[<index>] も使用可能です。このキーワードは、GET SOURCE メッセージのリストと同様の動作をします。インデックス番号は 0 から始まります。詳細については GET メッセージの説明を参照してください。

```
OPN 0x01000000
SET WIDTH 800
SET HEIGHT 600
END
HEX 0x0b000001
```

	GET WINDOWS BEG WND ACT 00ad2308 VIDEO3 END CLO 00ad2308 ACK OPN 0x01000000 SET WIDTH 800 SET HEIGHT 600 END HEX 0x0b000001 GET WINDOWS BEG WND ACT 00ad2308 VIDEO3 END CLO $W[0] ACK
形式	(コマンド: String, ソース ID: int)、(コマンド: String, 名前: String)、または(コマンド: String, var: String)
コマンド	OPN
返り値	選択が成功しなかった場合、エラーメッセージを返します。成功した場合は何も返しません。 「END」コマンドでメッセージグループを閉じると、メッセージまたはエラーメッセージでウィンドウハンドルを返します。 「RBK」コマンドでメッセージグループを閉じると、全ての変更はロールバックされ、ウィンドウは開かれません。
サンプル	OPN VIDEO1 **¥ea¥e5**¥00¥03¥4f¥50¥4e**¥e5**¥00¥06¥56¥49¥44¥45¥4f¥31**¥eb**¥3d
Cue サンプル	¥ea¥e1¥ff¥eb¥b4

Select source (SEL SRC) - オープンメッセージグループ

入力ソースを選択します。	
詳細	このメッセージは OPN コマンドと同様にソースを選択します。このコマンドの違いはソースをウィンドウに開かないという点です。このコマンドはソースのデフォルト値を取得します。
形式	(コマンド:String, サブコマンド:String)
コマンド	SEL SRC
返り値	GET メッセージの結果のみ返します。
サンプル	SEL SRC 0x01000000 GET WIDTH INT 800 END SEL SRC 000003E8¥r¥n **¥ea¥e5**¥00¥03¥53¥45¥4c**¥e5**¥00¥03¥53¥52¥43**¥e3**¥00¥00¥03¥e8**¥eb**¥3f
Cue サンプル	¥ea¥e1¥fe¥eb¥b4

Select window (SEL WND) - オープンメッセージグループ

表示中の任意のウィンドウを選択します。	
詳細	既存のウィンドウを選択し、ウィンドウのパラメータの設定または取得をするメッセージグループを開始します。 ウィンドウを選択するには、ウィンドウハンドル、$A、$W[<index>]、またはユーザー定義のキーワードを指定します。詳細については「OPN」メッセージの項目を参照してください。 $W[<index>] キーワードは、直近の GET WINDOWS コマンドを実行したときのウィンドウリストを使用します。インデックスは 0 から始まります。グループの本体メッセージには、GET と SET コマンドのみが含まれます。 SET コマンドの結果は、END でメッセージグループが閉じられたときに実行されます。
形式	(コマンド:String, サブコマンド:String)
コマンド	SEL WND
返信	GET メッセージの結果のみ返します。
サンプル	SEL WND 01000000 GET WIDTH INT 800 END SEL WND $W[0]¥r¥n **¥ea¥e5**¥00¥03¥53¥45¥4c**¥e5**¥00¥03¥57¥4e¥44**¥e5**¥00¥05¥24¥57¥5b¥30¥5d**¥eb**¥bf
Cue サンプル	¥ea¥e1¥fe¥eb¥b4

Begin information (BEG) - オープンメッセージグループ

サーバーからの応答メッセージの先頭に付くコマンドです。	
詳細	このメッセージは DXInterface からクライアントにのみ送信されます。 GET SOURCES、GET WINDOWS、UPD のリストで使用されます。
形式	(コマンド:String)
コマンド	BEG
返信	サーバーがこの応答メッセージをクライアントに送信します。
サンプル	BEG¥r¥n **¥ea¥e5**¥00¥03¥42¥45¥47**¥eb**¥8b

Close current selection (END) - クローズメッセージグループ

メッセージグループを閉じます。	
詳細	グループの先頭メッセージの種類によって動作が異なります。 OPN の後：選択したソースを開きます。 SEL WND / SEL SRC の後：現在のグループを閉じます。 BEG の後：リストの終わりを示します。
形式	(コマンド:String)
コマンド	END
返り値	メッセージグループの先頭メッセージによって異なります。
サンプル	END¥r¥n **¥ea¥e5**¥00¥03¥45¥4e¥44**¥eb**¥94

実行せずに現在の選択を閉じる (RBK)

処理を実行しないでメッセージグループを閉じます。	
詳細	メッセージグループで指定した先頭メッセージの種類によって動作が異なります。 OPN の後：変更操作を中止します。 SEL WIN、SEL SRC の後：現在のグループを閉じます。
形式	(コマンド:String)
コマンド	RBK
返り値	サーバーは ACK (acknowledge) メッセージを返します。
サンプル	RBK¥r¥n **¥ea¥e5**¥00¥03¥52¥42¥4b**¥eb**¥9c

値の取得 (GET)

詳細	このメッセージは 2 つのケースで使用できます。 ● 入力ソースのデフォルト値を取得する。 ● ウィンドウの現在のパラメータの値を取得する。 クライアントは、各 GET 要求に対する応答を判別するための「serial_id」を指定できます。serial_id フィールドはオプションです。
例	1. SEL SRC 00000001 GET WIDTH INT 800 END
	2. SEL WND $A GET WIDTH INT 800 END
形式	(コマンド:String, パラメータ名:String, serial_id:String)
コマンド	GET
パラメータ名	取得するパラメータの名前です。使用できるパラメータについては後述のパラメーター覧を参照してください。詳細な情報についてはユーザーズガイドとクイックスタートガイドを参照してください。
返り値	値またはエラーコードを返します。詳細については「Value message」を参照してください。値は文字列または int 型の整数値です。 テキストモードでは int 値はウィンドウハンドルの 16 進数コードで表されます。それ以外の場合は 10 進数の整数が使用されます。

値の設定 (SET)

パラメータの値を設定します。	
詳細	詳細については SEL WND と OPN コマンドを参照してください
形式	(コマンド:String, パラメータ名:String, 値:String) または (コマンド:String, パラメータ名:String, 値:int)
コマンド	SET
パラメータ名	設定するパラメータの名前です。利用可能なパラメータ、使用できるパラメータについては後述のパラメーター覧を参照してください。 ローカルまたはグローバル変数の指定も可能です。
返り値	成功の場合は ACK メッセージを返し、失敗の場合はエラーを返します。

パラメーター覧

	説明	Global	OPN	SEL WND	SEL SRC	値
ENCODING	返り値内の変数文字列の優先エンコーディングの設定	SET	-	-	-	ASCII, UCS2
MODED	現在のプロトコルバージョンの設定	SET	-	-	-	CUE, UNKNOWN
STRWIDTH	返り値の変数文字列の必要幅の設定	SET	-	-	-	0：任意の長さ その他の整数値は固定長の文字列で最後にスペースが入る
SOURCES	入力ソースのリストを返す	GET	-	-	-	value messages を参照
WINDOWS	ウィンドウのリストを返す	GET	-	-	-	value messages を参照
UPD	更新されたオブジェクトのリスト	-	-	-	-	value messages を参照
UPDATEPERIOD	セッションの UPD リクエストをシミュレートする期間の設定	SET	-	-	-	0: タイマーなし それ以外はミリ秒単位で期間を指定
E	最後に発生したエラーの可読メッセージを返す	GET	-	-	-	value messages を参照
RESULT	現在の OPN メッセージグループに result のユーザー定義キーワード名を設定	-	SET	-	-	大文字小文字を区別しない$A、$S...、$W...、$GLOBAL...を除く、$で始まる文字列値
ID	現在のオブジェクトの ID またはウィンドウハンドル	-	GET	GET	GET	0 ～ 0xFFFFFFFF
NAME	入力ソースの名前または指定されたウィンドウの入力ソース名	-	GET	GET	GET	value messages を参照
DUMP	すべてのウィンドウをパラメータ付きで WIL スクリプトとして一覧表示	GET	-	-	-	OPN ～ END コマンド QUIT コマンドで終了
LEFT	左上隅の水平方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
TOP	左上隅の垂直方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
WIDTH	ウィンドウの幅	-	SET/GET	SET/GET	GET	0 ～ 32767
HEIGHT	ウィンドウの高さ	-	SET/GET	SET/GET	GET	0 ～ 32767
CROP.HORBASE	水平方向のクロッピングの基準値。CROP.LEFT と CROP.RIGHT の値は、[0 ～ CROP.HORBASE-1] の範囲	-	GET	GET	GET	0 ～ 32767

	説明	Global	OPN	SEL WND	SEL SRC	値
CROP.VERBASE	垂直方向のクロッピングの基準値。CROP.TOP と CROP.BOTTOM の値は、[0 ～ CROP.VERBASE-1]の範囲	-	GET	GET	GET	0 ～ 32767
CROP.LEFT	表示領域の矩形の左上隅の水平方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
CROP.TOP	表示領域の矩形の左上隅の垂直方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
CROP.RIGHT	表示領域の矩形の右下隅の水平方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
CROP.BOTTOM	表示領域の矩形の右下隅の垂直方向の位置	-	SET/GET	SET/GET	GET	0 ～ 32767
SHOW	ウィンドウの状態	-	SET/GET	SET/GET	GET	0: MOTION 1: STILL 2: HIDE
TOPMOST		-	SET/GET	GET	GET	0, 1
ZOOM		-	SET/GET	GET/SET	GET	0 ～ 10
KEEPASPECTRATIO		-	SET/GET	GET/SET	GET	0, 1
HASFRAME		-	SET/GET	GET/SET	GET	0, 1
BRIGHTNESS		-	SET/GET	GET/SET	GET	0 ～ 16383
CONTRAST		-	SET/GET	GET/SET	GET	0 ～ 16383
HUE		-	SET/GET	GET/SET	GET	0 ～ 16383
SATURATION		-	SET/GET	GET/SET	GET	0 ～ 16383
SHARPNESS		-	SET/GET	GET/SET	GET	0 ～ 16383
GAMMA		-	SET/GET	GET/SET	GET	0 ～ 16383
COLORKEY		-	SET/GET	GET/SET	GET	0 ～ MAX_INT
VIDEOSTANDARD		-	SET/GET	GET/SET	GET	DXFG04 以降: TMDS: 116 't' RGB: 114 'r' Video: 118 'v' TMDSASense: 84 'T' RGBASense: 82 'R' VideoASense: 86 'V' その他の入力ソース: 110: NTSC 4:3 78: NTSC 16:9 112: PAL 4:3 80: PAL 16:9
SVIDEO		-	SET/GET	-	GET	0, 1
INSERTIONMODE		-	SET/GET	-	GET	0: 通常 1: 方フィールド 2: 両フィールド
AUTOENCODING	VNC プロパティ。 詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1

	説明	Global	OPN	SEL WND	SEL SRC	値
USE8BIT	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1
USECOPYRECT	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1
PREFERREDENCODING	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	VNC プロパティ。詳細はユーザーズガイド参照
VIEWONLY	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	SET/GET	GET	0, 1
DISABLECLIPBTRANSFER	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1
EMULATE3BUTTONS	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1
SWAPBUTTONS	VNC プロパティ。詳細はユーザーズガイド参照	-	SET/GET	-	GET	0, 1
HWACCELERATED	フレームグラバー用ハードウェアアクセラレーションモード。ハードウェアのリソースが必要	-	SET/GET	-	GET	0, 1
OPTIMIZATION	フレームグラバーの最適化。ビットマップ品質を選択	-	SET/GET	-	GET	0: 品質重視 1: 速度重視
MOTIONDETECT	入力ビデオ信号の動き検出	-	SET/GET	GET	GET	0, 1
ZORDER	ウィンドウの重なりの上下配置	-	-	SET	-	0: 下へ 1: 上へ
FONTS	システムで利用可能なフォントの一覧表示。詳細は OSD または OSD[index].FONTINDEX のパラメータを参照	GET	-	-	-	BEG STR <fontname> ... END

	説明	Global	OPN	SEL WND	SEL SRC	値
TIMEOUT	フレームグラバーのフレームタイム (単位：ミリ秒)	-	SET/GET	SET/GET	GET	TIMEOUT >= 10 の場合、ミリ秒単位のフレーム時間 0: ソースフレームレート 1: ソースフレームレート div1 2: ソースフレームレート div2 3: DXFG04 以降のデバイスの場合、ソースフレームレート div4
PHASE	サンプリングフェーズ	-	SET/GET	-	GET	0 ～ 16383
DEINTERLACER	デインターレースの有効 / 無効	-	SET/GET	-	GET	0, 1
DETECTTIMING	1 を設定すると、タイミングを検出	-	-	SET	-	1
DETECTPHASE	1 を設定すると、フェーズを検出	-	-	SET	-	1
DETECTHSTART	1 を設定すると、水平方向の開始値を検出	-	-	SET	-	1
PRESENT	指定したウィンドウの信号の有無を返す	-	-	GET	GET	0：信号あり 1：信号なし 2：未定義 (判別不能)
OSD.SIZE	選択したウィンドウの OSD ボックスの数。詳細は、「OSD」を参照	-	SET/GET	SET/GET	GET	0 ～ 16383
OSD[index].TEXT	OSD に表示する文字列	-	SET/GET	SET/GET	GET	index: 0 ～ 16383 text: 任意
OSD[index].FONTINDEX	GET FONTS テーブルにおける、選択したボックスのフォントインデックス	-	SET/GET	SET/GET	GET	index: 0 ～ 16383 value: 0 ～ 16383
OSD[index].FONT	OSD に表示する文字列のフォント	-	SET/GET	SET/GET	GET	text
OSD[index].X	指定した OSD ボックスの表示位置の X 座標 (横軸)	-	SET/GET	SET/GET	GET	0 ～ 16383

	説明	Global	OPN	SEL WND	SEL SRC	値
OSD[index].Y	指定した OSD ボックスの表示位置の Y 座標 (縦軸)	-	SET/GET	SET/GET	GET	0 ～ 16383
OSD[index].POINTSIZE	OSD に表示する文字列の大きさ	-	SET/GET	SET/GET	GET	0 ～ 255
OSD[index].COLOR.RED	OSD に表示する文字列の色 (赤)	-	SET/GET	SET/GET	GET	0 ～ 255
OSD[index].COLOR.GREEN	OSD に表示する文字列の色 (緑)	-	SET/GET	SET/GET	GET	0 ～ 255
OSD[index].COLOR.BLUE	OSD に表示する文字列の色 (青)	-	SET/GET	SET/GET	GET	0 ～ 255
OSD[index].BOLD	指定した OSD ボックスのフォントスタイル (太字)	-	SET/GET	SET/GET	GET	0, 1
OSD[index].ITALIC	指定した OSD ボックスのフォントスタイル (イタリック)	-	SET/GET	SET/GET	GET	0, 1
OSD[index].BACKGROUND	OSD の背景色	-	SET/GET	SET/GET	GET	0, 1
OSD[index].WNDCOORD	OSD 文字列の大きさをウィンドウサイズに比例して可変・固定を指定	-	SET/GET	SET/GET	GET	0, 1
AUDIOSOURCES	音声ソースのリストを取得	GET	-	-	-	value messages を参照
AUDIOCONNECTS	音声接続のリストを取得	GET	-	-	-	value messages を参照
HWMONALERT	ハードウェア・モニタの故障情報の取得	GET	-	-	-	value messages を参照
HWMONDATA	ハードウェア・モニタの正常情報の取得	GET	-	-	-	value messages を参照
FRAMECOLORSTATUS	入力のカスタムフレームの状態を取得	-			GET	文字列値 valid: 色あり invalid: 色なし
FRAMECOLOR	カスタムウィンドウフレームのカラー値を設定 / 取得	-		SET/GET		カラー値 : 0xbbggrr
TICKER.ENABLE	ティッカーステータスの設定 / 取得	SET/GET				0, 1
TICKER:DOCKED	ティッカーボックスをウォール下部に配置	SET/GET				0, 1

	説明	Global	OPN	SEL WND	SEL SRC	値
TICKER.LEFT	ティッカーボックスの左上隅の X 座標 (横軸)	SET/GET				0 ～ 16383
TICKER.TOP	ティッカーボックスの左上隅の Y 座標 (縦軸)	SET/GET				0 ～ 16383
TICKER.WIDTH	ティッカーボックスの水平方向の幅	SET/GET				0 ～ 16383
TICKER.HEIGHT	ティッカーボックスの垂直方向の幅	SET/GET				0 ～ 16383
TICKER.FONTINDEX	GET FONTS テーブルにおけるティッカーボックスのフォントインデックス	SET/GET				0 ～ 200
TICKER.POINTSIZE	ティッカーボックスのフォントサイズ	SET/GET				8 ～ 100
TICKER.BOLD	ティッカーボックスのフォントスタイル (太字)	SET/GET				0, 1
TICKER.ITALIC	ティッカーボックスのフォントスタイル (イタリック)	SET/GET				0, 1
TICKER.BGCOLOR.RED	ティッカーボックスの背景色 (赤)	SET/GET				0 ～ 255
TICKER.BGCOLOR.GREEN	ティッカーボックスの背景色 (緑)	SET/GET				0 ～ 255
TICKER.BGCOLOR.BLUE	ティッカーボックスの背景色 (青)	SET/GET				0 ～ 255
TICKER.TCOLOR.RED	ティッカーボックスのテキストカラー (赤)	SET/GET				0 ～ 255
TICKER.TCOLOR.GREEN	ティッカーボックスのテキストカラー (緑)	SET/GET				0 ～ 255
TICKER.TCOLOR.BLUE	ティッカーボックスのテキストカラー (青)	SET/GET				0 ～ 255
TICKER.SPEED	ティッカーテキストの移動速度	SET/GET				-20 ～ 20
TICKER.TEXT	ティッカーのテキスト	SET				文字列
TLBRIGHTNESS	QuadVideo プロパティ：左上 1/4 の輝度	-	SET/GET	SET/GET	GET	0 ～ 16383

	説明	Global	OPN	SEL WND	SEL SRC	値
TLCONTRAST	QuadVideo プロパティ：左上 1/4 のコントラスト	-	SET/GET	SET/GET	GET	0 ～ 16383
TLHUE	QuadVideo プロパティ：左上 1/4 のヒュー	-	SET/GET	SET/GET	GET	0 ～ 16383
TLSATURATION	QuadVideo プロパティ：左上 1/4 のサチュレーション	-	SET/GET	SET/GET	GET	0 ～ 16383
TRBRIGHTNESS	QuadVideo プロパティ：右上 1/4 の輝度	-	SET/GET	SET/GET	GET	0 ～ 16383
TRCONTRAST	QuadVideo プロパティ：右上 1/4 のコントラスト	-	SET/GET	SET/GET	GET	0 ～ 16383
TRHUE	QuadVideo プロパティ：右上 1/4 のヒュー	-	SET/GET	SET/GET	GET	0 ～ 16383
TRSATURATION	QuadVideo プロパティ：右上 1/4 のサチュレーション	-	SET/GET	SET/GET	GET	0 ～ 16383
BLBRIGHTNESS	QuadVideo プロパティ：左下 1/4 の輝度	-	SET/GET	SET/GET	GET	0 ～ 16383
BLCONTRAST	QuadVideo プロパティ：左下 1/4 のコントラスト	-	SET/GET	SET/GET	GET	0 ～ 16383
BLHUE	QuadVideo プロパティ：左下 1/4 のヒュー	-	SET/GET	SET/GET	GET	0 ～ 16383
BLSATURATION	QuadVideo プロパティ：左下 1/4 のサチュレーション	-	SET/GET	SET/GET	GET	0 ～ 16383
BRBRIGHTNESS	QuadVideo プロパティ：右下 1/4 の輝度	-	SET/GET	SET/GET	GET	0 ～ 16383
BRCONTRAST	QuadVideo プロパティ：右下 1/4 の輝度	-	SET/GET	SET/GET	GET	0 ～ 16383
BRHUE	QuadVideo プロパティ：右下 1/4 のヒュー	-	SET/GET	SET/GET	GET	0 ～ 16383
BRSATURATION	QuadVideo プロパティ：右下 1/4 のサチュレーション	-	SET/GET	SET/GET	GET	0 ～ 16383

	説明	Global	OPN	SEL WND	SEL SRC	値
PREVIEWSOURCES	プレビューソースのリストを取得	GET	-	-	-	value messages を参照
PREVIEW[n].ENABLE	ソース#n のプレビューの有効 / 無効。詳細は「PREVIEW」を参照してください。	GET/SET	-	-	-	0: 無効 1: 有効
PREVIEW[n].WIDTH	ソース#n のプレビュー画像の幅	GET/SET	-	-	-	120 ~ 768
PREVIEW[n].HEIGHT	ソース#n のプレビュー画像の高さ	GET/SET	-	-	-	90 ～ 576
PREVIEW[n].TIME	ソース#n のプレビュー画像の更新時間	GET/SET	-	-	-	500 ～ 60000 ms
PREVIEW[n].IMAGE	ソース#n のプレビュー画像のデータ	GET	-	-	-	png 画像の文字列

変数の削除 (RMV)

定義された変数をシステムから削除します。	
詳細	このコマンドは変数が存在しない場合や変数がすでに削除されていても、常に「success」を返します。 Remove コマンドは割り当てられたメモリを破棄するため、メモリリークを防ぐことができます。セッション置換子は、セッションが存在する場合において、TCP 接続が閉じられた際に自動的に削除されます。 このメッセージはシステムからグローバル置換子を削除する唯一の手段としてとても有用です。
形式	(コマンド:String, キーワード変数:String)
コマンド	RMV
返り値	成功の場合は ACK メッセージを返し、失敗の場合はエラーを返します。
サンプル	最初のセッションでウィンドウを開くことができます。 OPN 0x01000000 SET WIDTH 800 SET HEIGHT 600 SET RESULT $V1 END HEX 0x0b000001 SET $GLOBAL.V1 $V1 ACK もう一方のセッションで以下のコマンドを入力してウィンドウを閉じます。 ($V1 は不可視ですが、$GLOBAL.V1 は有効です) GET $V1 ERR 0x8000001D GET E STR Unknown window handle or variable! CLO $GLOBAL.V1 ACK RMV $GLOBAL.V1 ACK

アップデート要求 (UPD)

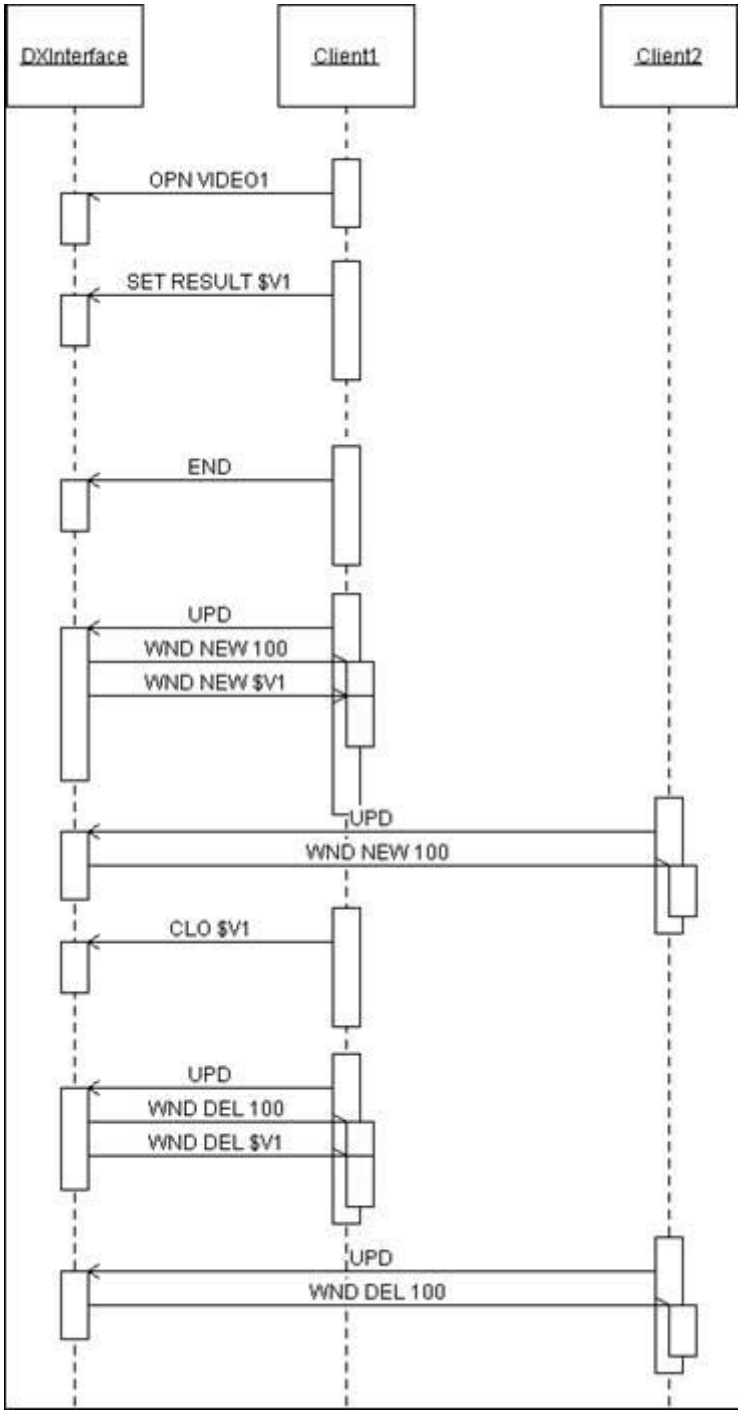
最後にこのメッセージを呼び出してからの変更点を返します。	
詳細	GET WINDOWS メッセージと同様の動作ですが、2 つの違いがあります。送信済みのウィンドウは送信されません。これらのウィンドウは ACT 状態にあります。 2 つ目の違いは、前回の UPD 呼び出し以降にソースリストが変更された場合、ウィンドウリストの変更後に STR SRC_UPD という特別なメッセージが送信されます。詳細については「GET SOURCES」または「Value messages」を参照してください。
提案	システムアーキテクチャは、クライアント制御で設計されており、これは通常、クライアントが要求を開始することになります。サーバー主導の更新を使用したい場合は、自動更新を要求する必要があります。これは、UPDATE_PERIOD 値を希望の更新時間 (ミリ秒) に設定することで可能です。

	「SET UPDATE_PERIOD 1000」というコマンドを使用することで、クライアントが 1 秒ごとに UPD を呼び出すのと同じように動作します。
形式	(コマンド: String)
コマンド	UPD
返り値	サーバーは前回の UPD コマンド以降の変更点のリストを、WND と STR のメッセージで返します。メッセージの最初と最後には、BEG と END が入ります。
サンプル	UPD¥r¥n **¥ea¥e5**¥00¥03¥55¥50¥44**¥eb**¥a6

Value messages (STR, INT, SRC, WND)

詳細	DXInterface サービスは、GET および UPD メッセージにバリューメッセージで応答します。**STR** (string 型)、**HEX** (int 型)、**INT** (int 型) のメッセージは、アトミックな値を送信します。	
	SRC	入力ソースを識別するために必要なすべての情報が含まれています。3 バイトの「SRC」キーワード、16 進数のソース ID、STRWUDTH 値で制限可能なソース名で構成されています。
	WND	ウィンドウを識別するために必要なすべての情報が含まれています。3 バイトの「WND」キーワード、状態キーワード、16 進数のウィンドウハンドルで構成されています。状態は以下の 4 つがあります。 **NEW**：クライアントに初めて通知されるウィンドウです。 **ACT**：ウィンドウが開いた状態で、クライアントにすでに通知されています。 **DEL**：ウィンドウが削除されたことを表します。 **UPD**：ウィンドウが閉じられた後、同じウィンドウハンドルで再び開かれた場合に使用されます。この状態が使われることは稀です。

一般的なウィンドウのライフサイクルは以下のようになっています



どのクライアントも RESULT 変数を設定することで、ビデオウィンドウを開くことができます。
RSULT 変数を使用したウィンドウはそのクライアント固有のウィンドウとみなされ、クライアントにはウィンドウ ID だけでなく、ID の代わりにユーザー定義のキーワードを含む別のメッセージが通知されます。
これらのキーワードはプライベートであるため、他のクライアントにはウィンドウハンドルのみが通知されます。返される文字列値の最大幅は、STRWIDTH の値に依存します。詳細については「SET」メッセージの項目を参照してください。

タイミング検出 (DTI)

指定ウィンドウの信号のタイミング検出を開始します。	
詳細	パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	DTI
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	DTI $W[0]¥r¥n

位相検出 (DPH)

指定ウィンドウで自動位相検出を開始します。	
詳細	パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	DPH
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	DPH $W[0]¥r¥n

ウィンドウを閉じる (CLO)

指定したウィンドウを閉じます。	
詳細	パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	CLO
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	CLO $W[0]¥r¥n

ウィンドウの強制終了 (KIL)

指定したウィンドウを強制終了します。	
詳細	ウィンドウの強制終了は、ウィンドウの種類がアプリケーションの場合でも即座に実行されます。 パラメータはウィンドウハンドルで、8 文字の 16 進数値です。

	直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	KIL
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	KIL $W[0]¥r¥n

接続解除の通知 (DISC)

詳細	システムは DISC 切断コマンドを送信する場合があります。 これは非同期なので応答コマンドはありません。 DXInterface はクライアントにこのメッセージを送信し、接続が解除され、今後利用できなくなることを通知します。 その後、クライアントは再接続の確認のために PING メッセージを定期的に送信できます。再度利用可能になると、サーバーは PING メッセージを送信します。
形式	(コード:byte)
コード	0F1h
サンプル	DISC¥r¥n **¥ea¥e5**¥00¥04¥44¥49¥53¥43**¥eb**¥e1
Cue サンプル	¥ea¥e1¥f1¥eb¥a7

OSD の操作

詳細	ビデオウィンドウに OSD パラメータを設定できます。OSD はウィンドウ上の個別のテキストボックスで構成されます。 各ボックスはテキストのほかに、フォント、位置情報のパラメータで構成されています。詳細についてはパラメータ名の表を参照してください。 ボックスは全てのウィンドウでインデックスが付与されます。インデックスは 0 から始まります。「SET OSD.SIZE」コマンドで OSD のサイズ数を設定し、リストの終わりに項目を追加、または消去ができます。
サンプル	新しく開かれたビデオウィンドウに 2 つの OSD テキストを追加 GET FONTS BEG STR ARIAL STR TIMES END OPN VIDEO1 SET OSD.SIZE 2 SET OSD[0].TEXT VIDEO SET OSD[0].FONT ARIAL SET OSD[1].TEXT A1 SET OSD[1].COLOR.RED 255 END HEX 0xAAAAAAAA

	既存のウィンドウに 2 つの OSD テキストを追加 SEL WND $V1 SET OSD.SIZE 2 SET OSD[0].TEXT VIDEO SET OSD[0].FONT ARIAL SET OSD[1].TEXT A1 SET OSD[1].COLOR.RED 255 END

3.5.1. Z 順序の操作

ビデオウィンドウの重なりの順序を変更するには、以下のコマンドを使用します。

最背面に移動 (ZLO)

指定したウィンドウの Z 順序を最背面に移動します。	
詳細	このメッセージはウィンドウの topmost (常に前面) 設定が解除されます。 パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	ZLO
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	OPN 0x01000000 SET RESULT $GLOBAL.V1 SET LEFT 0 SET TOP 0 END HEX 0x0036f960 OPN 0x01000002 SET RESULT $GLOBAL.V2 SET LEFT 0 SET TOP 0 END HEX 0x0036fd18 ZLO $GLOBAL.V2 ACK 開いたウィンドウは他のウィンドウの下に表示されます。

最前面に移動 (ZRA)

指定したウィンドウの Z 順序を最前面に移動します。	
詳細	Top Most が有効になっている他のウィンドウがある場合は、そのウィンドウの下に位置付けされます。 パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(コマンド:String, ハンドル:int) または (コマンド:String, キーワード変数:String)
コマンド	ZRA
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	OPN 0x01000000

	SET RESULT $GLOBAL.V1 SET LEFT 0 SET TOP 0 END HEX 0x0036f960 OPN 0x01000002 SET RESULT $GLOBAL.V2 SET LEFT 0 SET TOP 0 END HEX 0x0036fd18 ZRA $GLOBAL.V1 ACK 開いたウィンドウは他のウィンドウの上に表示されます。

Top Most の切り換え (ZTO)

指定したウィンドウの Top Most (常に前面) 設定の有効 / 無効を切り換えます。	
詳細	パラメータはウィンドウハンドルで、8 文字の 16 進数値です。 直近に送信した GET WINDOWS メッセージの結果リストのウィンドウインデックスである $W[x] でも代用できます。キーワード $A を使用すると、セッションで最後に開いたウィンドウのハンドルに置き換わります。また、オープンメッセージグループの SET RESULT コマンドを使用したキーワード指定も可能です。
形式	(command:String, handle:int) または (command:String, substitutor:String)
コマンド	ZTO
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合や、ウィンドウが見つからない場合にはエラーメッセージを返します。
サンプル	OPN 0x01000000 SET RESULT $GLOBAL.V1 SET TOPMOST 1 END HEX 0x0036f960 ZTO $GLOBAL.V1 ACK ウィンドウの topmost 設定が有効から無効に切り換わります。

3.5.2. 音声の操作

音声入力ソースの取得

コマンド	GET AUDIOSOURCES
返り値	現在の入力音声の一覧を返します。GET SOURCES コマンド実行時の結果と似ていますが、音声のある入力ソースのみが含まれます。本機が音声処理に対応していないか、または入力音声が構成されていない場合は、空のリストになります。
サンプル	GET AUDIOSOURCES BEG SRC 0x0a000000 “Universal 1” SRC 0x0a000001 “Universal 2” SRC 0x0a000002 “Universal 3” SRC 0x0a000003 “Universal 4” SRC 0x0a010000 “HDMI 1” SRC 0x0a010001 “HDMI 2” END

音声の出力状態の取得

コマンド	GET AUDIOCONNECTS
返り値	入力音声のモニタへの出力状態をリストアップします。リストの形式は、以下のサンプルを参照してください。
サンプル	GET AUDIOCONNECTS BEG AUD 0x0a000000 1 AUD 0x0a010000 2 AUD MUTE 3 AUD MUTE 4 END 上記の例の場合、Universal1 の音声がスクリーン 1 に、HDMI1 の音声がスクリーン 2 に出力されています。スクリーン 3 とスクリーン 4 はミュートとなっています。

音声の割り当て

出力チャンネルに音声チャンネルを割り当て、スクリーンから音声を出力します。	
形式	(コマンド:String, ソース ID:int, 出力インデックス:int) もしくは (コマンド:String, ソース ID:String, 出力インデックス:int) もしくは (コマンド:String, ソース ID:int, 出力インデックス:String) もしくは (コマンド:String, ソース ID:String, 出力インデックス:String))
コマンド	AC **MUTE**：コマンド「AC」の後ろに MUTE を指定することで、スクリーンの音声出力がオフになります。 **ALL**：出力インデックスに ALL を指定することで、選択した入力音声を全ての出力に割り当てます。
返り値	サーバーは ACK メッセージを返します。パラメータが解決できない場合は、エラーメッセージを返します。
サンプル	AC 0x0a000000 1 AC 0x0a010000 2

	AC MUTE 3 AC MUTE 4 ACK 上記コマンドで出力 1 に Universal 1 のオーディオ、出力 2 に HDMI1 のオーディオ、出力 3 と出力 4 にはミュートが割り当てられます。 以下の記述でも上記と同じ動作になります。 AC MUTE ALL AC 0x0a000000 1 AC 0x0a010000 2 ACK

3.5.3.ハードウェアモニタの操作

ハードウェアアラートの取得

故障が疑われるハードウェアの情報を一覧表示します。	
詳細/サンプル	システムの状態を監視する際に役立ちます。 数分間隔でこの問い合わせコマンドを送信し、結果を確認するのが効果的です。ただし、この方法だけでは不十分な場合もあります。システムメンテナンス時には、より詳細なハードウェア情報が必要になります。その場合は、次に説明するコマンドを活用してください。
形式	(コマンド:String)
コマンド	GET HWMONALERT
返り値	サーバーはハードウェアの一覧を返します。メッセージの最初と最後には、BEG と END が入ります。ハードウェアセンサーが異常値を検出した場合、そのハードウェアのデータが表示されます。何も問題がない場合は、空のリストになります。 表示される値の意味は、以下の通りです。 ● Type ○ HWMF: ファン ○ HWMP: 電力 ○ HWMT: 温度 ○ HWMS: 電源ユニット ● Index: センサーの識別子 ● Device name: デバイスを識別する文字列値 ● Device number: デバイスを識別する数値 ● Sensor name: センサー名の文字列値 ● Minimal value: 有効値の下限 ● Maximum value: 有効値の上限 ● Current value: 実際の測定値 測定値の単位は以下の通りです。 ファン： RPM (Rounds Per Minite) 電力： mV (ミリボルト) 温度： ℃ (摂氏) 電源ユニット： 1 = 正常 / 2 = 異常
サンプル	GET HWMONALERT BEG HWMF 11 D1B8-1616 0 “Fan – Chassis 1” 350 2147483647 0 END 上記の例では、ファンセンサーが異常を検出したことを通知しています。

ハードウェアモニタのデータを取得

ハードウェアモニタのデータベースの更新を要求します。	
詳細	HWMU (as HardWare Monitor Update)コマンドは、ハードウェアモニタのデータベースを照会する「GET HWMONDATA」コマンドの前に送信するようにデザインされています。
形式	(コマンド:String)
コマンド	HWMU
返り値	サーバーは ACK メッセージを返します。
サンプル	HWMU ACK GET HWMONDATA BEG **HWMS 0 D1B8-1616 0 “PS (R) - AC” 1 1 1** **HWMS 1 D1B8-1616 0 “PS (R) - POWER” 1 1 1** **HWMP 2 D1B8-1616 0 “3.3 V” 2970 3630 2988** **HWMP 3 D1B8-1616 0 “1.2 V (A)” 1080 1320 1197** **HWMP 4 D1B8-1616 0 “5 V (A)” 4500 5500 5026** **HWMP 5 D1B8-1616 0 “5 V (B)” 4500 5500 5000** **HWMP 6 D1B8-1616 0 “1.2 V (B)” 1080 1320 1183** **HWMP 7 D1B8-1616 0 “1.8 V” 1600 2000 1783** **HWMP 8 D1B8-1616 0 “1 V” 900 1100 1040** **HWMP 9 D1B8-1616 0 “12 V” 10800 13200 12250** **HWMT 10 D1B8-1616 0 “Zone Crossbar” 0 70 64** **HWMT 11 D1B8-1616 0 “Zone chassis” 0 50 28** **HWMF 12 D1B8-1616 0 “Fan − Crossbar” 800 2147483647 858 HWMF 13 D1B8-1616 0 “Fan − Chassis 1” 350 2147483647 1412** **HWMF 14 D1B8-1616 0 “Fan − Chassis 2” 350 2147483647 1371** **HWMF 15 D1B8-1616 0 “Fan − Chassis 3” 350 2147483647 1354** **END**

3.5.4. 入力ソースのプレビュー

プレビューソースの一覧を取得

詳細	本機のプレビュー機能が有効な場合、以下のコマンドを使用することで、プレビュー可能な入力チャンネルの一覧を表示できます。
形式	(コマンド:String)
コマンド	GET PREVIEWSOURCES
返り値	プレビュー機能が有効な入力チャンネルの一覧を返します。メッセージの最初と最後には、BEG と END が入ります。プレビューが有効な入力チャンネルがない場合は、空のリストになります。
サンプル	GET PREVIEWSOURCES BEG SRC 0x0a008000 “HDMI 1” SRC 0x0a008001 “HDMI 2” SRC 0x0a008002 “HDMI 3” SRC 0x0a008003 “HDMI 4” SRC 0x0a000000 “Universal 1” SRC 0x0a000001 “Universal 2” SRC 0x0a000002 “Universal 3” SRC 0x0a000003 “Universal 4” END

プレビューソースのパラメータ操作

入力チャンネルのプレビューを開始、または停止します。	
詳細	コマンドには、ソースのインデックス番号を指定する必要があります。上記サンプルの一覧では、先頭の hdmi 入力 (ソース ID: 0x0a008000) は、インデックス「0」で識別され、最後の Universal 入力 (id: 0x0a000003) は、インデックス「7」で識別されます。
形式	(コマンド:String)
コマンド	PREVIEW[n].ENABLE
返り値	GET: プレビューの状態 (0 = 無効 / 1 = 有効) を返します。 SET: ACK メッセージを返します。
サンプル	SET PREVIEW[0].ENABLE 1 ACK

プレビュー画像のサイズを取得、または変更します。	
詳細	コマンドには、ソースのインデックス番号を指定する必要があります。
形式	(コマンド:String)
コマンド	PREVIEW[n].WIDTH PREVIEW[n].HEIGHT
返り値	GET: int 型の整数値を返します。 SET: ACK メッセージを返します。
サンプル	GET PREVIEW[0].WIDTH INT 160 GET PREVIEW[0].HEIGHT INT 90

プレビュー画像の更新時間を取得、または変更します。	
詳細	コマンドには、ソースのインデックス番号を指定する必要があります。
形式	(コマンド:String)
コマンド	PREVIEW[n].TIME
返り値	GET: int 型の整数値を返します。(単位：ミリ秒) SET: ACK メッセージを返します。
サンプル	SET PREVIEW[0].TIME 5000 ACK

プレビュー画像の取得

プレビュー画像の更新時間を取得、または変更します。	
詳細	コマンドには、ソースのインデックス番号を指定する必要があります。 プレビュー画像を取得するには、指定した入力チャンネルのプレビューが有効になっている必要があります。「PREVIEW[n].ENABLE」コマンドを参照してください。プレビューを有効にして画像が取得できる状態になるまで、数秒間待つ必要があります。 プレビュー画像は、TIME パラメータに応じて更新されます。2 つのフレーム間の経過時間はミリ秒単位で定義されます。 画像の更新は、GET PREVIEW[0].IMAGE とは独立して実行されますが、TIME パラメータは、GET PREVIEW[0].IMAGE を再実行する間隔と同じ値に設定することをお勧めします。 【注意】この操作では、大量のデータを転送することになります。TIME、WIDTH、HEIGHT パラメータは、適切な値を設定する必要があります。
形式	(コマンド:String)
コマンド	GET PREVIEW[n].IMAGE
返り値	最初の 4 バイトは必ず「PNG」になります。5 バイト目から画像データが始まります。1 文字が 1 つの 16 進数を表し、2 文字で png データの 1 バイトの 16 進数を表しています。
サンプル	GET PREVIEW[0].IMAGE PNG 89504e470d0a1a0a0000000d49484452000000a000000078080200000fac06e3 8000000017352474200aece1ce900000004674 14d410000b18f0bfc61050000000 97048597300000ec300000ec301c76fa8640000004f49444154785eedc1010d000 000c2a0f74f6d0e37200000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000e0 540de1780001d3194edb0000000049454e44ae426082 データの先頭 16 文字が、png シグネチャを表しています。 (89 504E47 0D0A 1A 0A)

3.6. エクスプレッションと演算子

WIL コマンドで使用できるエクスプレッションのルールは以下の通りです。

1. WIL コマンドのエクスプレッションは ＝ で始まります。

2. SET と GET コマンドの中で使用できます。SEL WND または OPN は必要ありません。それらは独立して処理します。

3. 演算子の優先順位は以下となります。これらは C 言語に似ていますが、同じではありません。

ブール演算子:	&& ||
比較演算子:	== != <= >= < >
集計演算子:	+ -
積演算子:	* / % | & ^
単項演算子:	~ ! + - (?:)

アトミックタームはバイナリ (0b110001) 、16 進数 (0xFEFF) 、10 進数 (15) です。
評価式は置換子 ($mask) または後述の関数です。

4. 利用可能な関数は以下となります。
   - **#WNDEXSR[<src>]**
     ソース ID (または置換子でも可) で指定したウィンドウが 1 つでも存在する場合、1 を返します。
   - **#WNDBYSR[<src>].<param>**
     <src> ソース ID (または置換子でも可) で検索された最初のウィンドウのパラメータ <param> を返します。
   - **#WNDEXID[<wh>]**
     指定したウィンドウハンドルを持つウィンドウが存在する場合、1 を返します。
   - **#WNDBYID[<wh>].<param>**
     <wh> で指定されたウィンドウのパラメータ <param> を返します。
   - **#WNDEXNM[<srcnm>]**
     ソース名 (または置換子でも可) で検索されたウィンドウが 1 つでも存在する場合、1 を返します。
     検索条件： X==Y ::= UPPER(X).STARTWITH(UPPER(Y))
   - **#WNDBYNM[<srcnm>].<param>**
     <srcnm> ソース名で検索された最初のウィンドウのパラメータ <param> を返します。
     検索条件： X==Y ::= UPPER(X).STARTWITH(UPPER(Y))
   - **#SRCEXID[<id>]**
     指定された ID を持つソースが存在する場合、1 を返します。
   - **#SRCBYID[<id>].<param>**
     <id> で識別されるソースのデフォルトパラメータ <param> を返します。
   - **#SRCEXNM[<srcnm>]**
     ソース名 (または置換子でも可) が存在する場合、1 を返します。
     検索条件： X==Y ::= UPPER(X).STARTWITH(UPPER(Y))

- **#SRCBYNM[<srcnm>].<param>**
  名前で検索されるソースのパラメータ <param> を返します。
  検索条件： X==Y ::= UPPER(X).STARTWITH(UPPER(Y))

5. 使用例

```
//The value of the expression is 1:
get =((1+5)*2)/8

//Decodes $a
get=($a==0?10:($a==1?15:21))

//Returns the name of the first RGB source
get =#SRCBYID[0x03000000].NAME

//Returns the name of the source named PAL1
get =#SRCBYNM[PAL1].NAME

//Returns the brightness of the first RGB source
set $src =#SRCBYID[0x03000000].NAME
get =#WNDBYSR[$src].BRIGHTNESS

//Returns whether the first RGB source is present
set $src =#SRCBYID[0x03000000].NAME
get =#WNDBYSR[$src].PRESENT
```

4. APPENDIX. チュートリアル

4.1. コマンドラインからのウィンドウのオープン、クローズの方法

DXInterface の TCP インターフェースを通して映像を開くことが可能です。dxlink.exe と呼ばれるアプリケーションを使用して、DXInterface へ標準入力のコマンドを送信します。
DXConfig でビデオソースの名前を「TEST1」として設定します。

まず、C:¥temp¥open.vbs という名前のファイルを以下の内容で作成してください。

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objTextFile = objFSO.OpenTextFile("a.out.txt", 2, True)
strLF = Chr(13)
objTextFile.Write "OPN TEST1" & Chr(13)
objTextFile.Write "SET RESULT $GLOBAL.INSTANCE1" & Chr(13)
objTextFile.Write "SET LEFT 100" & Chr(13)
objTextFile.Write "SET TOP 200" & Chr(13)
objTextFile.Write "SET WIDTH 720" & Chr(13)
objTextFile.Write "SET HEIGHT 480" & Chr(13)
objTextFile.Write "END" & Chr(13)
objTextFile.Write "QUIT" & Chr(13)
objTextFile.Close
Set WshShell = WScript.CreateObject("WScript.Shell")
rem WshShell.Run "cmd /c dxlink.exe 127.0.0.1 -p 6466 <a.out.txt",0
WshShell.Run "cmd /c dxlink.exe 127.0.0.1 -p 6466 <a.out.txt",0
```

スクリプトは a.out.txt というファイルを作成し、それを dxlink プログラムの助けを借りて TCP/IP 経由で DXInterface に送信します。

ファイルは全てコマンド行で構成されています。OPN 行の TEST1 文字列は、開きたい映像ソースの名前を選択します。SET RESULT コマンドは識別子を設定し、あとでウィンドウを閉じる際に使用します。その他の SET コマンドは、デフォルトのウィンドウパラメータを変更します。
END コマンドの送信が成功すると、ウィンドウが開かれます。QUIT コマンドは、接続を解除し DXLINK アプリケーションを終了します。

次に、前述のウィンドウを閉じるコマンドファイル作成します。内容は以下となります。
C:¥temp¥close.vbs という名前のファイルを以下の内容で作成してください。

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objTextFile = objFSO.OpenTextFile("a.out.txt", 2, True)
strLF = Chr(13)
objTextFile.Write "CLO $GLOBAL.INSTANCE1" & Chr(13)
objTextFile.Write "QUIT" & Chr(13)
objTextFile.Close
Set WshShell = WScript.CreateObject("WScript.Shell")
rem WshShell.Run "cmd /c dxlink.exe 127.0.0.1 -p 6466 <a.out.txt",0
WshShell.Run "cmd /c dxlink.exe 127.0.0.1 -p 6466 <a.out.txt",0
```

コマンドを受け付けるように Dxinterface を設定します。

1. [スタートメニュー]→[DXInterface Configuration]を開きます。
2. メニューの[Configuration]を開きます。
3. Global タブを開きウォールの IP アドレスを設定します。本機で使用している場合、127.0.0.1 を使用します。
4. Apply を押してください。
5. TCP タブを開きます。
6. 「Enable TCP Communication」を有効にします。
7. TCP ポートを 6466 に設定します。
8. プロトコルを「Text WIL」に設定します。
9. OK を押してください。

以下を Windows の「ファイル名を指定して実行」に入力すると、ビデオウィンドウを開くことができます。
c:¥temp¥open.vbs

以下を Windows の「ファイル名を指定して実行」に入力すると、ビデオウィンドウを閉じることができます。
c:¥temp¥close.vbs

4.2. テキストプロトコルを使った手動でのウィンドウの開閉

以下のコマンドを使用することで、ビデオウィンドウを開き、それを管理することが可能になります。
以下の例では、DXConfig で SRC1 と名付けられたビデオソースを開くことができます。

```
OPN SRC1
SET RESULT $GLOBAL.MYVIDEO1
SET WIDTH 800
SET HEIGHT 600
END
```

END コマンドを入力した直後にウィンドウが表示されます。END コマンドを実行するまで SET コマンドで指定した値はウィンドウに反映されません。キーワード $GLOBAL.MYVIDEO1 を使用することで、後でこのウィンドウを参照できます。
このコマンドで、デフォルトのウィンドウサイズが 800 x 600 に変更されます。

ウィンドウのパラメータの変更は SEL WND コマンドで行います。

```
SEL WND $GLOBAL.MYVIDEO1
SET WIDTH 1024
SET HEIGHT 768
END
```

END コマンドでウィンドウの大きさ(幅、高さ) の変更を実行します。GET と SET コマンド参照は、利用可能なすべてのフィールド (brightness、pausing、frame visibility など) が含まれています。
詳しくは 3.4 プレゼンテーション層 - テキストプロトコル (68 ページ) を参照してください。

CLO コマンドを使ってこのウィンドウを閉じることができます。

```
CLO $GLOBAL.MYVIDEO1
```

4.3. ウィンドウレイアウトを自動的に WIL スクリプトに取り込む

以下の内容のスクリプトファイル (C:¥dump.txt) を作成してください。
各行の後で Enter を押してください。

```
GET DUMP
QUIT
```

DXWallControl のディレクトリから以下のコマンドラインを実行してください。

```
dxlink.exe <c:¥dump.txt >c:¥layout.txt
```

レイアウトは、QUIT コマンドで終わる一連の OPN コマンドとして、c:layout.txt に保存されます。このレイアウトは、以下のコマンドを実行して復元することができます。

```
dxlink.exe <c:¥layout.txt
```

【注意】このコマンドは DXInterface で利用可能なパラメータのみを保存します。
このコマンドはいくつかの内部プロパティや、ウィンドウの重なり (Z) 順序を保存しません。ウォールの完全なレイアウトを保存したい場合は、DXWallControl シナリオを使用してください。

4.4. 利用可能なソースのリストアップ

以下のコマンドを使用することで、利用可能な映像ソースのリストを表示します。

```
GET SOURCES
```

4.5. 管理ウィンドウのリストアップ

以下のコマンドを使用することで、管理ウィンドウのリストを表示します。

```
GET WINDOWS
```

管理ウィンドウとは、本機で制御された起動中、一時停止中、または非表示状態にあるビデオウィンドウやアプリケーションウィンドウを指します。

4.6. ウォールのクリア

以下のコマンドを使用することで、ウォール上に表示中のウィンドウをすべて閉じます。

```
CLR
```

4.7. ステップを実行しても何も表示されない場合

本項は、全ログレベルが有効にした際に以下のログが表示されるが、ウォールに希望するレイアウトが表示されない、といった場合に有効です。

```
>>EVT EVT1
DXWallControl found: C:/scenarios/scenario1.dxs suffix:_c22d451
Event launched: EVT1_ c22d451
```

このような場合のときは、以下の手順を試してください。

1. シナリオファイル (C:/scenarios/scenario1.dxs) に、外部イベントコード EVT1 を持つ有効なステップがあるかどうかを確認してください。
2. シナリオを変更する場合は、まず DXWallControl に保存し、DXInterface configurator のメニューある「Pause All」と「Resume All」を順番に実行してください。
3. DXInterface には、応答時間を短縮し、システム負荷を軽減するためのシナリオキャッシュが含まれています。Pause / Resume の手順は、DXInterface のシナリオキャッシュを更新し、変更したファイルを強制的に読み込ませます。DXInterface Configuration ダイアログの Apply または OK ボタンを押した際も同様の効果があります。
4. ソフトウェアのバージョンを確認してください。DXWallControl アプリケーションを手動で実行した場合、目的のステップを実行できることを確認してください。
5. 本機を再起動し、再度確認してください。それでも問題解決しない場合は営業部までご連絡ください。

4.8. DXInterface から本機の電源オフと再起動

インストール CD には、DXInterface の「電源オフ」および「再起動」操作をサポートする 3 つのバッチファイルが含まれています。
これらのバッチファイルは CD の「¥Support¥shutdown」フォルダにあります。ファイルは以下の通りです。

- poweroff.bat
  システムの電源を切ります。
- reboot.bat
  本機を再起動します。
- abort.bat
  poweroff.bat または reboot.bat によって開始された処理を中止します。

DXInterface での電源オフ / 再起動操作の方法

1. **アプリケーションソース**として「poweroff.bat」と「reboot.bat」を設定します。(詳細は下記参照)
2. 「poweroff.bat」と「reboot.bat」のアプリケーションソースを開く 2 つのシナリオステップを作成します。(詳細は下記参照)
3. DXInterface にて、これらのシナリオステップを他のステップと同じ方法で実行します。

詳細：

「poweroff.bat」、「reboot.bat」バッチファイルをアプリケーションソースとして設定する

- インストール CD からバッチファイル「poweroff.bat」、「reboot.bat」、「abort.bat」を、本機の任意のフォルダにコピーしてください。例：C:¥
- DXConfig を開き、「Input Sources」タブに移動します。
  1. Source Type ペインで「Application」を選択します。Source ペインで空欄の行を右クリックで選択し、「Configure」を選択してください。
  2. 開いたダイアログで、「Application source」として設定したいプログラムを選択できます。
  3. 「Application」の欄に、バッチファイルのフルパスを入力します。例：C:¥poweroff.bat 「Arguments」と「Start in」は空欄のままにしておきます。
  4. 「Console Application」と「No Console Window」のチェックボックスを両方とも選択します。
  5. 次に「Launch」ボタンを押すと、設定したバッチファイルが起動します。バッチファイルがシステムをシャットダウンしようとしますが、25 秒間の猶予があります。シャットダウンの手順をキャンセルする「abort.bat」をこの時間内に実行してください。「Launch」ボタンを押す前に、「abort.bat」バッチファイルを実行する用意をしておいてください。
  6. Configuration ダイアログの OK ボタンを押します。同様の手順にて「reboot.bat」バッチファイルも設定します。
  7. DXConfig ダイアログの OK ボタンを押して、設定を終了します。

シナリオステップの作成

1. DXWallControl を起動します。本機のローカルで作業する場合でも、今回は「Local DXWallControl」は使用しないでください。
2. オプションメニューで「Execute Scenario」の選択を解除します。
3. 正しく「poweroff.bat」と「reboot.bat」が設定されていれば、ソースリストの中にそれらが表示されているはずです。シナリオステップの作成は、ドラッグアンドドロップで行います。「poweroff」ソースをドラッグし、DXWallControl ウィンドウのシナリオ部分の空白エリアにドロップします。そうすることで、新しいステップが作成されます。
4. 同様の手順にて「reboot」ソースもステップに追加します。
5. DXInterface にて、これらのシナリオステップを他のシナリオステップと同じように扱うことができるようになります。

DXN6000 シリーズ 取扱説明書

<コマンドガイド>

Ver.1.0.0

発行日：2021 年 9 月 21 日

株式会社アルバニクス

本　　社　〒242-0021 神奈川県大和市中央 7-9-1
TEL: (046) 259-6920
FAX: (046) 259-6930
E-mail: info@arvanics.com
URL: http://www.arvanics.com