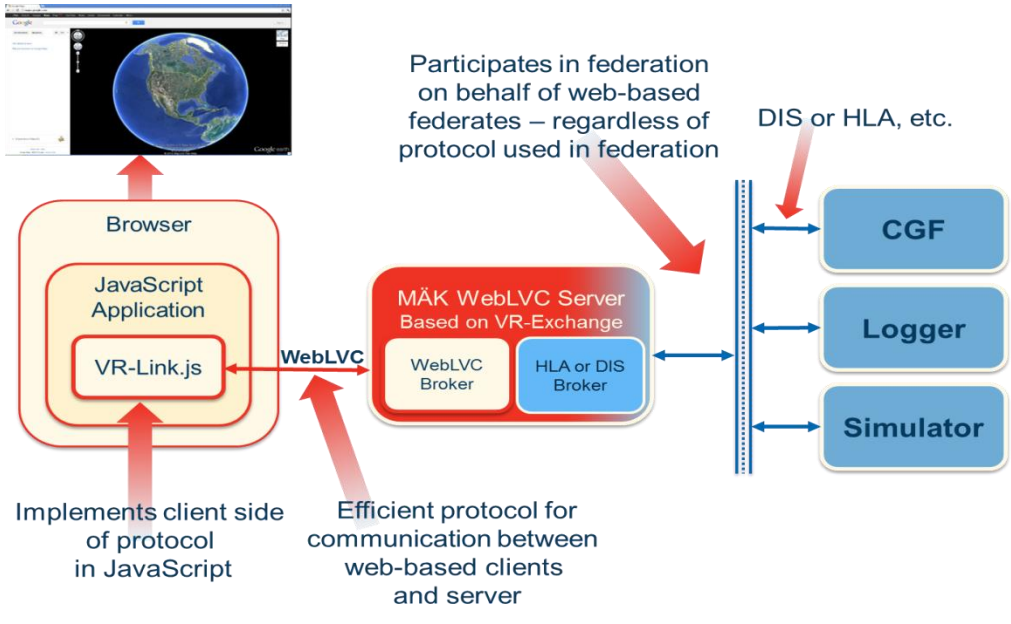
MÄK WebLVC Server 1.2 Release Notes

Introduction

MÄK WebLVC Server is the linchpin that enables web-based simulation to interoperate with traditional distributed simulation federations. Web-based client applications connect to MÄK WebLVC Server using WebSockets, and exchange information with the server using the WebLVC Protocol. MÄK WebLVC Server translates simulation data between the WebLVC Protocol and traditional M&S protocols such as DIS and HLA.

MÄK WebLVC Server is based on MÄK's VR-Exchange interoperability platform, which supports many-to-many translation between various interoperability protocols through the use of "Brokers" for each protocol. When you install MAK WebLVC Server, you are essentially installing a copy of VR-Exchange pre-configured with a Broker for DIS and a Broker for the WebLVC protocol.



In the diagram above, the federates on the right may be using DIS, HLA 1.3, HLA 1516, HLA Evolved, TENA, or any other protocol for which a VR-Exchange Broker exists. See documentation for more information on the VR-Exchange architecture, and on the configuration of various Brokers.

New Features and Updates

MÄK WebLVC Server 1.2 contains the following new features and updates:

- New apps for WebLVC Suite:
  - **Commander**: Command VR-Forces entities from a tactical map interface.
  - **Instructor**: Control and stimulate simulation with events for training.

    - **VR-Vantage Controller**: Control a VR-Vantage Stealth viewer from a mobile device or desktop.
    - **VR-Vantage Viewer**: View live video streaming from VR-Vantage.

  WebLVC Suite is a separately-priced configuration of WebLVC Server that includes several additional web apps. The WebLVC Suite apps have more specialized capabilities for observing and controlling a simulation. Contact sales@mak.com for more information on WebLVC Suite.
- Supports for mobile devices – the new WebLVC Suite apps include support for running on tablets and iPhones. Commander, Instructor, VR-Vantage Controller, and VR-Vantage Viewer work on mobile devices as well as the desktop.
- Support for new VR-Forces message extensions to WebLVC Protocol. These messages provide the capability to remote control VR-Forces, and include messages for sending tasks, text reports, changing environment conditions, creating objects, and VR-Forces simulation engine status. See the Commander app for a working example of these messages in context.
- Support for new VR-Vantage messages extensions to WebLVC Protocol. These messages provide the capability to manipulate the observer and enable/disable various visual effects such as sensor modes, model scaling, radio communication lines, etc. See the Instructor and VR-Vantage Viewer apps for examples of these messages in context.
- Source files are now included for all of the web apps, located in the clientSource folder. (The html folder contains the built and deployed versions of the web apps.) See the section "Building the Client Web Apps" below for instructions on how to build the apps.

Documentation Update

- The procedure for configuring the automatic FOM translation example app (Web Bounce) has been updated and simplified. See section "Automatic FOM Translation Example: Web Bounce" below for details.

Supported Platforms

Operating System	Version
Windows XP Windows 7	Microsoft Visual C++ 10.0 (64 bit)

MAK Product Compatibility

Product	Version	Notes
VR-Link	4.0.9b	
MAK RTI	4.2	Minimum version required for automatic FOM translation (see "Extending WebLVC with Automatic FOM Translation below.")
VR-Forces	4.2	Version required for use with Close Air Support, Commander, and Instructor WebLVC Suite apps.
VR-Vantage	1.6	Version required for use with Instructor, Commander

	VR-Vantage Controller, and VR-Vantage Viewer apps. Note that version VR-Vantage 1.6.1 is ***not*** compatible with WebLVC Server 1.2.

Browser Compatibility

Browser	Version	Notes
Chrome	Recent version (2014)	
Firefox	Recent version (2014)	VR-VantageController not supported on Firefox.
Safari	Recent version (2014)	
Internet Explorer	10, 11	Internet Explorer 9 and earlier does not support web sockets. 2D-3D Viewer app not supported on Internet Explorer. VR-VantageController not supported on InternetExplorer.

Installation

To install MÄK WebLVC Server, simply run the MÄK WebLVC Server installer, and follow the prompts.

Included in the installation

Base Installation

- The MÄK WebLVC Server software itself – pre-configured with Brokers for DIS, HLA, and WebLVC
- A set of client-side libraries called vrlink.js to help you implement the client-side of the WebLVC protocol in your web applications
- A set of JavaScript example web apps, with source code, to get you started in building your own web-based client applications. The base example web apps consist of:
    - Simple Listener
    - Simple Publisher
    - Messages Inspector
- A draft of the WebLVC Specification (0.2).

WebLVC Suite Installation

The WebLVC Suite Installation includes the Base Installation plus the WebLVC Suite apps, consisting of:

- Commander
- Instructor
- VR-Vantage Controller
- VR-Vantage Viewer
- 2D-3D Viewer
- Close Air Support
- Detonate Now
- VR-Forces Control

The installer also launches two additional 'add-ons' installers for VR-Forces and VR-Vantage. These launchers install data and plug-ins necessary in order to work in conjunction with some of the web apps, including Close Air Support, Commander, Instructor, VR-Vantage Controller, and VR-Vantage Viewer. The installers for VR-Forces add-ons and VR-Vantage add-ons are located in base folder of the WebLVC Server installation:

```
<WebLVC Server installation directory>/VR-ForcesWebLVCAdd-ons.exe
<WebLVC Server installation directory>/VR-Vantage1-6WebControllerAdd-ons.exe
```

These installers can be run at any time, wherever your VR-Forces and VR-Vantage installations are located.

Getting Started

To try out MÄK WebLVC Server:

1. Launch MÄK WebLVC Server from the Start Menu: Programs -> MÄK Technologies -> MÄK WebLVC Server x.y.z -> WebLVC Server. The standard VR-Exchange GUI should come up, allowing you to see that there are brokers running for the DIS and WebLVC protocols.
2. Start the Client Apps page from the Start Menu: Programs -> MÄK Technologies -> MÄK WebLVC Server x.y.z →WebLVC Client Apps. This should launch your default web browser, and automatically load the HTML file that contains links to all of the example client apps.
3. Click on the Simple Listener thumbnail image to launch the app.
4. At this point, you should see a simple Web-based map application running in your browser. But you will not see any entities; because you aren't running any DIS federates yet.
5. Launch any DIS application that publishes entity data. For example, you can run the VR-Link F-18 example application, VR-Forces, or a MAK Data Logger configured to play back a sample Logger file.
6. You should now see entities listed in the MÄK WebLVC Server GUI to indicate that there is data coming into the Server from the HLA or DIS side, and you should see entity icons being displayed in the web browser overlaid on the 2D map.

Important Notes:

- In order for an HLA Broker to connect to an HLA federation, you will need to have an HLA Run-time Infrastructure (RTI) installed. You can download the MÄK RTI for free here: http://www.mak.com/products/link-simulation-interoperability/hla-rti-run-time-infrastructure.html. MÄK RTI version 4.2 or later is recommended.
- If you do not have an HLA RTI, MÄK WebLVC Server will report that it cannot connect to an HLA federation, but you will still be able to interoperate with DIS federates.
- MÄK WebLVC Server fully supports deployments on closed networks behind firewalls, with no internet access. Browser-based federates will be able to communicate with traditional DIS/HLA federates running on the same network, through MÄK WebLVC Server. However, our sample Simple Listener application requires an internet connection in order to access the terrain and imagery data that it displays on the map. If you are trying out MÄK WebLVC Server for the first

time on a machine without access to the internet, we recommend that you run the Message Inspector sample web application, so that you can at least see that data is flowing from DIS/HLA federates to your browser through MÄK WebLVC Server. Recognize that it is entirely possible to build web-based PVDs, Stealth viewers, maps applications, etc. without requiring internet access. But you would need a locally deployed terrain and imagery server such as ArcGIS Server, Google Enterprise Server, or MÄK's VR-TheWorld Server in order to serve the data.

- By default MÄK WebLVC Server launches the WebLVC Broker (configured to use port 9101) and the DIS broker (configured to use port 3000). Additional brokers such as the HLA Evolved Broker (configuration dictated by the RTI) can be added at runtime and configured to launch automatically.

Configuring the Brokers

- The MÄK WebLVC Server documentation (accessible in the .\doc directory or from the Start menu) covers how to configure the DIS and HLA brokers for your environment.
- By default, the web socket server comes up on port 9101. To change to a different port, you can right-click on the WebLVC Broker from the main VR-Exchange GUI, and choose Configuration. When the WebLVC Broker configuration dialog comes up, change the WebSocket Port parameter.
    - When you click apply, the WebLVCBroker.bcf broker configuration file is written to the bin directory.

Relationship between MÄK WebLVC Server and Your Web Server

In the Getting Started example above, we relied on the built-in web server capability of WebLVC Server to serve the client app to the browser. In a real deployment, you will likely want your web applications to be served by a web server such as Apache or IIS – especially if your web applications exist within the context of a web site.

To try this out, copy the html folder from the MÄK WebLVC Server installation directory to the folder in which your web server *typically looks for HTML files (e.g. C:\inetpub\html for IIS, and C:\Program Files\Apache\html for Apache). Our* html folder contains the MÄK WebLVC Server example client applications, along with the vrlink.js libraries that the applications use to implement the WebLVC protocol. You should now be able to bring up a web browser, and browse to http://yourwebserver/ExampleWebApps/index.html, where yourwebserver is the URL of your web server. (ExampleWebApps/index.html is the location of the main page for launching all of the client apps).

Note that the approach above assumes that you will be running MÄK WebLVC Server on the same machine as your web server, because our sample applications are hard-coded to initiate WebLVC (WebSocket) connections to the same machine that served the HTML web page containing the application. If you would like to run MÄK WebLVC Server on a separate machine (not your web server), please contact support@mak.com, and we can walk you through the configuration process.

Note also, that our sample apps are hard-coded to initiate their WebLVC (WebSocket) connections on port 9101 – since that is the port used by default in MÄK WebLVC Server. If you want to use a different port, you will need to change the port used by the server as described above, and then pass that port

number t our sample applications via URL parameters. For example, enter the following URL in your browser: http://yourwebserver/SimpleListener/index.html?wsPort=9202.

For those who working in network environments that allow traffic only on port 80, we do support web proxying, which allows clients to use port 80 for both their static HTTP traffic and the WebLVC WebSocket traffic. If you are interested in such a configuration, please contact support@mak.com, and we can walk through the configuration process.

Extending WebLVC with Automatic FOM Translation

The MÄK WebLVC Server currently has built-in translators to convert the message types contained in the current WebLVC Protocol Specification between WebLVC and either DIS or HLA (e.g. Entity, Aggregate, Fire, and Detonate). But MÄK WebLVC Server can also automatically generate extensions to the WebLVC protocol based on the information contained in an HLA FOM, and automatically perform translation between HLA and those WebLVC extensions.

In other words, MÄK WebLVC Server is self-extending: MÄK WebLVC Server reads the names and data types of classes, attributes, parameters, structures and enumerations directly from an HLA FOM, and maps them to corresponding JSON elements. (No such similar capability exists for WebLVC <-> DIS translation, because there is no DIS equivalent to a FOM).

The MÄK WebLVC Server's Automatic FOM Translation capability relies on VR-Exchange's HLA Broker's "generic" or pass-through translation capability. When an HLA Broker is configured to perform generic translation, attributes and parameters of various FOM classes will pass through the VR-Exchange portal in "raw" form, without the HLA Broker converting the data into any kind of VR-Exchange protocol-independent representation. Once the raw values get across the portal, the WebLVC Broker can import a FOM document to understand how to interpret the data, and can convert the binary data to the WebLVC JSON representation.

To use this Automatic FOM Translation capability, you must do the following:

1) Configure your HLA Broker to choose which object and interaction classes should be sent across the VR-Exchange portal using generic translation. For a detailed explanation of setting up generic translation, see the VR-Exchange Users Guide.

2) Configure your WebLVC Broker with the name of the file that defines your FOM data. This is either an HLA-1.3-compliant OMT file or the xml-based file from HLA 1516 or HLA Evolved. Set the value of the "omtFiles" variable in your WebLVCBroker.bcf configuration file to point to the desired FOM document. The default FOM file is "VR_Link.omt". You can define more than one FOM file when using HLA Evolved modules, by separating the files using a semi-colon. For example: vrlink.xml;vrlinkextensions.xml;vrforcesextensions.xml.

Mapping HLA Data Representations to WebLVC Data Representations

The Automatic FOM Translation capability maps HLA types to JSON/WebLVC objects according to the following scheme:

The following HLA types translate to a JSON string (e.g. "HelloWorld")

- string
- HLAcString
- HLAstring
- HLAcount
- handle
- timeType

The following HLA types translate to a JSON number string (e.g. "1234")

- HLAbyte
- HLAfederateHandle
- HLAinteger16BE
- HLAinteger16LE
- HLAinteger32BE
- HLAinteger32LE
- HLAinteger64BE
- HLAinteger64LE
- HLAmsec
- HLAoctet
- HLAoctetPairBE
- HLAoctetPairLE
- HLAseconds)
- HLAunicodeChar
- UnsignedInteger16BE
- UnsignedInteger32BE
- UnsignedInteger64BE
- UnsignedShort
- VerfierIntegerTime
- int
- long
- long long
- octet
- short short
- unsigned int
- unsigned long
- unsigned long long
- unsigned short

The following HLA types translate to a JSON floating point string (e.g. "1234.5")

- HLAfloat32BE
- HLAfloat32LE
- HLAfloat64BE
- HLAfloat64LE
- double
- float

The following HLA types translate to a single JSON character (e.g. "a")

- HLAASCIIchar

char

The HLA Boolean type translates to a JSON Boolean string (e.g. “False”)

HLA enumerations map to a JSON string name:

```
var BoolEnum = "HLAFalse"
```

HLA arrays translate to arrays of corresponding JSON types,

```
var arrayOfLongs = [ "1","2","3" ]
```

Composite types are composed from primitive JSON types and other composite types, for example:

```
var messageLocation = {
    X : "-2696500",
    Y : "-4430500",
```

No table of contents entries found. `};`

```
var messageToPublish = {
   MessageKind: 1,
   ObjectType:
       "ObjectRoot.BaseEntity.PhysicalEntity.Platform.Aircraft",
   ObjectName: "From Web",
   WorldLocation: messageLocation
};
```

Automatic FOM Translation Example: Web Bounce

The MÄK WebLVC Server includes an example named Web Bounce that uses Automatic FOM Translation to translate a simple FOM. This example is based on the HLA Bounce example that comes with the MAK RTI (see section B.5 “HLA Bounce” in the MAK RTI Reference Manual for details on this example).

The Web Bounce example demonstrates how to configure a simple FOM to be generically translated to WebLVC. The HLA Bounce example application publishes one or more bouncing ball objects in an HLA federation. The Web Bounce web client app listens for and displays these published objects in a web browser.

To run the Web Bounce example:

1. Start MÄK WebLVC Server .
2. Load the WebLVC Client Apps page.
3. Launch the Web Bounce example located on the Client Apps page.
4. Follow the instructions for configuring the server and the HLA Bounce app on that page and then click on launch button there to start the Web Bounce app.

In the Web Bounce app federate window, you should see a mirror image of the bouncing balls being published by the HLA Bounce example federate.

You can also examine the JSON-based WebLVC extensions that are auto-generated from the MakHLABounce1516e FOM by opening the Message Inspector client app, available from the start menu under MAK Technologies→MAK WebLVC Server→WebLVC Client Apps.

Building the Client Web Apps

The MÄK WebLVC Server installation includes the source code and project files needed to build the web apps. Source files are located in the clientSource folder. (Pre-built deployed versions of those apps are located in the html folder.)
We use the following Open Source tools for frontend development. These tools include the following:

- Yeoman (yeoman.io) for scaffolding web apps
- Bower (bower.io) for dependency management
- Grunt (gruntjs.com) for automating the build process

Some of the simpler apps do not make use of the complete build process, but the more sophisticated apps make full use of this tool-chain. The procedure for installing and using these tools to build the apps is described below.

Prerequisites

Install the following packages. Install the latest versions available, unless otherwise noted. Node.js, Git, Ruby, and Python must all be in your PATH.

Node.js http://nodejs.org/download/

Git http://git-scm.com/download/win

Ruby http://rubyinstaller.org/downloads/

Compass (Install Ruby first and ensure it is in your PATH)
In a DOS window, type the following to install compass:

```
gem install compass
```

Note: Ruby/Compass is necessary for those apps that depend on Compass/SASS in generating CSS style sheets (e.g. vrlinkJS2/apps/ExampleWebApps).

Set up the global development environment

This step only needs to be performed once. It installs Yeoman, Grunt, Bower, and Yeoman Angular generator globally.

Open windows command prompt and run:

```
npm install -g yo
npm install -g generator-angular
```

Set up a project environment

Each project requires a one-time set-up step, in which we retrieve the development tools dependencies and the 3rd party application dependencies. In this example, we'll set up DetonateNow.

Open a command prompt. In the apps/DetonateNow folder, type:

```
npm install
bower install
```

The npm command installs the tools you need to build the project, such as the grunt tools for image compression, concatenation, Compass, etc.

The bower command installs the 3rd party application dependencies you've set up for the project (e.g. vrlinkJS, AngularJS, jQuery, etc.).

The project is now ready to be extended, built, or run.

Run the Project (locally)

To run the project using Grunt's built-in local webserver:

1. cd to apps\DetonateNow
2. run grunt serve(r)

This will start a web server and launch the DetonateNow app in a browser. It also runs a watch task and will refresh the page anytime a change is made to the source code.

Build the Project for Deployment

To build a version of the project suitable for deployment (minified, concatenated, etc.):

1. cd to apps\DetonateNow
2. run grunt build

This will minimize images, html, concatenate the JavaScript files, and place the results in a new dist folder. The dist folder contains everything the web app needs for deployment. At this point you can rename and copy the dist folder to deploy the example to the web server of your choice.

Bug Fixes

- 2D-3D Viewer app now works on the latest version of Firefox (version 29).

Known Problems

- 2D-3D Viewer app does not work on latest version of Chrome (34.0.1847.131 m) or Internet Explorer 10/11.
- VR-VantageController does not work on latest version of Firefox (29) or Internet Explorer 11.

- The sensor mode and ‘look’ controls in the Instructor app do not work in HLA. The other controls (weather, time of day, saved views) do work in HLA.