Yada 2.1.4

OPERATORS MANUAL

2008-01-26

TABLE OF CONTENTS

1 What's in the package?

The Yada 2.1.4 archive contains all files required to operate the Yada communication system. It also includes example configurations, documentation, source code, necessary libraries, licenses for 3rd party open source modules and a sound fix for Linux systems. The files are organized in folders **alsa**, **data**, **dist**, **doc**, **footswitch**, **licenses**, **src** and **peripherallink.** The contents of some of the more significant ones are described below.

1.1 data

The files in the **data** folder are used to set up and configure the system. Files outputted from the system are also stored here together with resources needed by the system. The data folder is divided into six subfolders; **configurations**, **exercises**, **logs**, **properties**, **recordings** and **resources**. The files in the resources folder should not be modified. The logs and recordings folders contain system log files and recorded audio. The files in the configurations and exercises folders are described in detail later.

1.2 alsa

This folder contains a fix that makes it possible to run the system under Linux without a soundcard that supports hardware mixing. This fix is needed due to a bug in the java sound routines that opens and allocates a hardware sound output device directly. Instructions are found in this folder.

1.3 dist

The **dist** folder contains yada2.1.3.jar which is the main file for the system and all system dependent java libraries. Java version 6 is required to run the system.

1.4 footswitch

This folder contains platform dependent libraries for the **footswitch**. These libraries must be installed in order to use the footswitch. There are two options, using an ftdi chip or serial com-port. Detailed installation instructions are found in the folder.

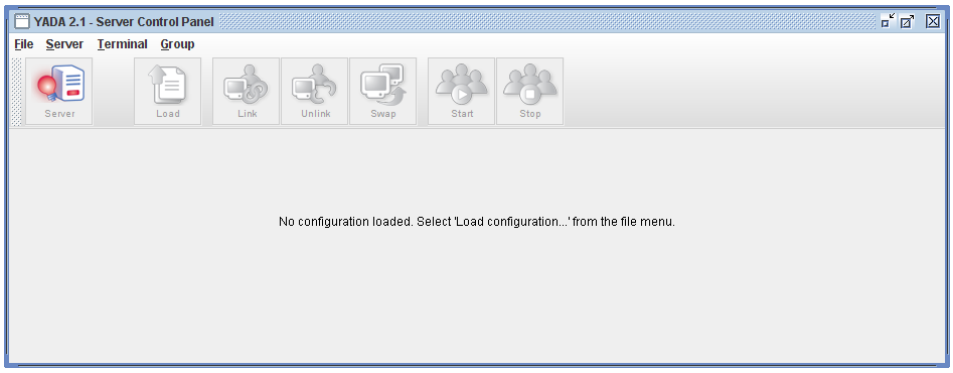
2 Setting up the server

Open the file **data/properties/serverproperties.xml** in a text editor and check the HTTP and UDP ports. Activate event logging on the server side by setting the LogEvents property to true. Events include radio, phone and group activity. Each group has its own log file that is re-created every time that the group is started. The logs are stored in **data/logs** and are never deleted.

Start a terminal and go to the the root directory of the package. Start the server by typing:

```
java -jar dist/yada.jar -s [ENTER]
```

The server control panel should now be visible on the screen as shown in the figure below. The control panel is used to control the communication system, for example to load configuration files and to start and stop the server. See chapter 5 for details.



Try running the application without the -s switch to see a list of the different execution modes:

```
-s      Start the server
-c      Start a client terminal
-c id   Start a client terminal with a specified id
-l      List sound devices
-d      Test sound devices
-f      Test the foot-switch
-m      Test the microphone level
```

3 Configuring the system

The system is configured by editing a configuration xml file describing available terminals, groups, radio channels, roles and players. Derive the new file from the configuration template found in **data/configuration/configuration-template.xml** when preparing the customized system configuration. One simple and another more advanced example are found in the same folder. The different sections in the configuration file are described below.

3.1 Configuration

This is the root node of the configuration document. It should specify the **codec** attribute as:

```
codec="null|jspeex:1-10"
```

where **null** indicates that no codec is used and raw data is sent over the network. The value **jspeex** indicates that the java port of the speex codec is used to compress the voice packets (www.speex.org). The value after the colon indicates the quality of speech, higher value means better quality and larger data packets.

```
<Configuration codec="jspeex:7">
```

3.2 TerminalDefs

The Terminal nodes define the physical terminals in the communication system. Each Terminal node must specify the following attributes:

```
id="1-4000"

x="0-"

y="0-"
```

where the **x** and **y** attributes specify where the terminal is located in the control panel view and should correspond to where the terminal is physically located in the room. The unit for these values are screen coordinates.

```
<TerminalDefs>
    <Terminal id="1" x="0" y="0">
        <Name>Terminal</Name>
    </Terminal>
    ...
</TerminalDefs>
```

3.3 GroupDefs

The Group nodes define the available groups in the system. Each Group node must specify the **id** attribute as:

```
id="1-4000"
```

The Color tag specifies the color of the group visible only in the server control panel. Search the web for ”hex color” to find a simple color value generator.

```
<GroupDefs>
    <Group id="1">
        <Name>Group</Name>
        <Color>#BBBBFF</Color>
    </Group>
    ...
</GroupDefs>
```

3.4 ChannelDefs

The Channel nodes define the available channels in the system. Each Channel node must specify the **id** attribute and may specify the following other attributes:

```
id="1-4000"

groupid="1-4000"

state="rxtx|rx|off"

locked="true|false"

hidden="true|false"

recordable="true|false"

showgroup="true|false"
```

The **groupid** attribute is used to bridge channels between groups, see Bridging groups section below for details. The **state** attribute specifies the initial state of the radio channel, the **locked** attribute indicates whether the state can be changed by the user or not. A **hidden** channel is invisible in the terminal application. A channel that has the **recordable** attribute set can record incoming audio and **showgroup** means that the group name is appended to the channel name in the terminal application which is useful when bridging channels.

```
<ChannelDefs>
    <Channel id="1">
        <Name>Channel</Name>
    </Channel>
    ...
</ChannelDefs>
```

3.5 RoleDefs

The Role nodes define the different roles that can be played. Each Role node should specify the **id** attribute and may specify the following other attributes:

```
id="1-4000"

groupid="1-4000"

showgroup="true|false"
```

The **groupid** attribute is used to bridge channels between groups, see Bridging groups section below for details. The **showgroup** attribute specifies whether the group name is appended to the role name in the terminal application or not. This is useful when bridging channels.

Each Role node must specify a list of phone destinations, i.e. references to roles that can be reached from the defined role's interphone.

A RoleRef node without the **id** attribute specified indicates that an empty slot should be placed at this location in the terminal application.

```
<RoleDefs>
    <Role id="1">
        <Name>Role</Name>
        <PhoneSetup>
            <RoleRef id="x"/>
            ...
        </PhoneSetup>
    </Role>
    ...
</RoleDefs>
```

3.6 PlayerDefs

The Player nodes define the players in the system. Each Player node must specify the **id** attribute as:

```
id="1-4000"
```

Each Player node must specify a list of channel references in ChannelSetup. Each ChannelRef node is allowed to override each of the attributes specified in the ChannelDefs section above. The attributes on the xRef nodes always have a higher priority than the definitions.

Each Player node must also specify a list of played roles in RoleSetup. All attributes can be overridden on the RoleRef node as well, but since the only two attributes allowed on the Role/RoleRef nodes (except for **id**) are **groupid** and **showgroup** the explanation is continued in the Bridging groups section below.

A RoleRef node without the **id** attribute specified indicates that an empty slot should be placed at this location in the terminal application.

```
<PlayerDefs>
    <Player id="1">
        <Name>Player</Name>
        <ChannelSetup>
            <ChannelRef id="1"/>
            ...
        </ChannelSetup>
        <RoleSetup>
            <RoleRef id="1"/>
            ...
        </RoleSetup>
    </Player>
    ...
</PlayerDefs>
```

3.7 Bridging groups

When creating a configuration without paying any attention to group bridging all players are fully isolated to their assigned group. The same channel and role definitions may be used in different groups, played by different persons but each of these channels and roles are still separated from the other groups.

Bridging groups is a simple way of allowing cross-group intercommunication. This can be useful in many cases, for example when a single supervisor must be able to communicate with all players independent of the group they are assigned to.

Keep in mind that the attributes on xRef nodes have the highest priority, the attributes on the nodes in the xDefs sections have medium priority and the own group id has the lowest priority.

Example 1: There are three groups configured, G1-G3. All players in these groups have access to channel C1, which is similar in definition but separate when used in different groups. Player P1 must be able to use channel C1 to communicate with all these groups. The ChannelSetup for P1:

```
<ChannelSetup>
    <ChannelRef id="1" groupid="1" showgroup="true"/>
    <ChannelRef id="1" groupid="2" showgroup="true"/>
    <ChannelRef id="1" groupid="3" showgroup="true"/>
</ChannelSetup>
```

Example 2: There are three groups configured, G1-G3. There must be one radio channel, C2, that is common for all groups. A common group, G4, is set up between existing groups to manage the common channel C2. In the example the common channel should also be locked in state **rxtx**. The Channel definition for channel C2:

```
<Channel id="2" groupid="4" state="rxtx" locked="true">
        <Name>C2</Name>
</Channel>
```

Example 3: All players, independent of which group they are assigned to, must be able to call the supervisor, P1, using the interphone. A new supervisor group, G9, is defined. The **groupid** attribute is specified on the Role node to override all players own group id.

```
<Role id="1" groupid="9">
    <Name>Supervisor</Name>
    <PhoneSetup>
        ...
    </PhoneSetup>
</Role>
```

4 Setting up a client terminal

A terminal is configured by editing the xml file **data/properties/terminalproperties.xml**. The contents of this file is explained in this chapter.

4.1 Selecting sound devices

The first step in the setup procedure is to select proper sound input and output devices. List available sound devices by typing:

```
java -jar dist/yada.jar -l [ENTER]
```

First a list of all installed devices is presented and then a list of output and input devices with the required capabilities. Select one output device and one input device and move on to the sound device test:

```
java -jar dist/yada.jar -d <out_dev> <in_dev> [ENTER]
```

First a single ring tone is heard to test the output device, then two different sounds are played simulaneously to test mixing capabilities. Finally the microphone input is looped back to the output through the system to test full duplex. It is possible to try out a number of different parameters using this test. To get a full list of test modes, type:

```
java -jar dist/yada.jar -d [ENTER]
```

The input level of the microphone can be visualized by running the application with the -m switch:

```
java -jar dist/yada.jar -m <in_dev> [ENTER]
```

4.2 Terminal setup

```
<entry key="TerminalId">4</entry>
<entry key="UserInterfaceStyle">full</entry>
<entry key="HiddenMouseCursor">false</entry>
```

`TerminalId`

The unique id for the terminal. This id must refer to one of the terminals defined in the configuration under the TerminalDefs node. Must be a value between 1 and 4000. This value is overridden if the terminal is started with a specified id parameter.

`UserInterfaceStyle`

The style of the terminal user interface; **full** for fullscreen user interface taking over the entire screen, **slim-horiz** for a slim horizontal user interface and **slim-vert** for slim vertical user interface. Default is **full**.

`HiddenMouseCursor`

Hides the mouse cursor in full style if this value is set to **true**. Use together with touch

screen monitors. Default is **false**.

4.3 Sound system setup

```
<entry key="SoundOutputDevice">0</entry>
<entry key="SoundInputDevice">0</entry>
<entry key="JitterBufferSize">2</entry>
<entry key="OutputBufferSize">2.4</entry>
<entry key="InputBufferSize">2.4</entry>
<entry key="SignalVolumeAdjustment">0.8</entry>
```

`SoundOutputDevice / SoundInputDevice`

The id's of the sound devices selected in section 4.1.

`JitterBufferSize`

The size of the jitter buffer in number of packets. Smaller jitter buffer decreases latency between sender and receiver but makes the system more sensitive to network packet disturbances. Set to a value between 1 and 20. Default is 4.

`OutputBufferSize / InputBufferSize`

The size of the output and input sound buffer in packets. One packet is always 20ms. Smaller buffers decreases latency but makes the system more sensitive to variations between sound card timer and system timer. Set to a decimal value between 1 and 4, default is 3.5.

`SignalVolumeAdjustment`

An adjustment value for the sound signal volume. Default is 1, meaning the volume is unchanged. Set to 0.8 to get 80% of the volume in the file.

4.4 Footswitch setup

```
<entry key="FootSwitchInterface">ftdi</entry>
<entry key="FootSwitchInvertCTS">false</entry>
<entry key="FootSwitchPollTime">200</entry>
```

`FootSwitchInterface`

The connection interface of the foot switch. Set this value to **ftdi** to use the devices based on the FTDI chip. To use a foot switch connected to a com port, set this value to the port to which the device is connected. On a Windows system this value is set to **COMx**, where **x** is the port number. On a Linux system the value is set to **/dev/ttyUSBx** or similar. Default is **ftdi**.

`FootSwitchInvertCTS`

The CTS signal is used to indicate whether the footswitch is pressed or not. Some footswitches uses an inverted CTS signal. Set this value to true to invert the CTS signal in the system. Default is **false**.

`FootSwitchPollTime`

The yada client polls the CTS signal with a constant period. Decrease this value to get a faster response time for the footswitches. A lower value may increase system load slightly. Default is 200 ms.

4.5 Client setup

Most values in this section only has to be modified in rare cases. Leave them to their default values if possible.

```
<entry key="MulticastAddress">224.0.23.61</entry>
<entry key="MulticastPort">36608</entry>
<entry key="MulticastTTL">3</entry>
<entry key="LocalBindAddress">null</entry>
<entry key="LocalBindPort">0</entry>
<entry key="PeripheralLinkPort">36612</entry>
```

`MulticastAddress`

The multicast ip address on the client. All clients that should be able to communicate through the multicast socket must have the same multicast address. Valid addresses range from 224.0.0.0 to 239.255.255.255. Default is **224.0.23.61**, which is an unallocated multicast address according to IANA (www.iana.org).

`MulticastPort`

The multicast port on the client. All clients that should be able to communicate through a multicast socket must have the same multicast port. Default is **36608**.

`MulticastTTL`

The Time-To-Live value for multicast packets in number of hops. This is the range of the multicast packets in the network. Multicast packets sent from a client will pass through the specified number of routers before it is discarded. Larger value makes it possible to find more clients when searching for multicast hosts but will increase the network load. Default is **3**.

`LocalBindAddress`

The local ip address to bind the UDP socket to. This is useful if multiple network cards are installed on the terminal computer. In most cases this value should be set to **null** which is also the default.

`LocalBindPort`

The local port to bind the UDP socket to. The default value **0** uses any available local port.

`PeripheralLinkPort`

The port on the yada klient to which the simulator connects. When connected, the yada client sends messages about radio sessions. Default port is **36612**, set to zero to disable this feature.

4.6 Server setup

```
<entry key="ServerAddress">192.168.10.112</entry>
<entry key="ServerHttpPort">36600</entry>
<entry key="ServerUdpPort">36604</entry>
```

`ServerAddress`

The hostname or IP address of the server.

`ServerHttpPort`

The TCP/HTTP port on the server. Default is **36600**.

`ServerUdpPort`

The UDP port on the server. Default is **36604**.

4.7 Color setup

The entries in the color section specify all indicator colors in the terminal application. The default value for all entries is light grey. Search the web for ”hex color” to find a simple color value generator.

4.8 Starting up the terminal

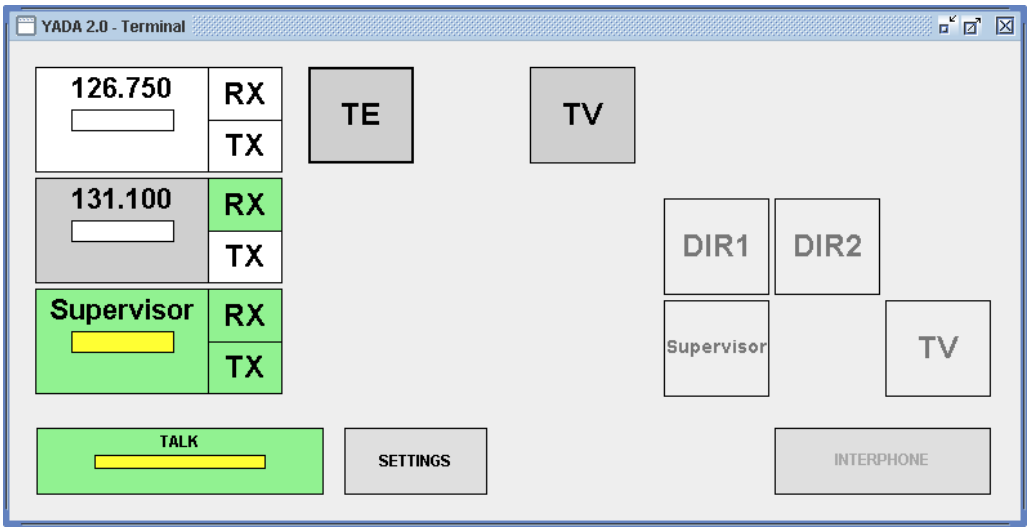
The terminal is started by running:

```
java -jar dist/yada.jar -c [ENTER]
```

or optionally with the terminal id specified on the command line as:

```
java -jar dist/yada.jar -c <term_id> [ENTER]
```

which is convienient if the same terminalproperties.xml is used with many terminals.

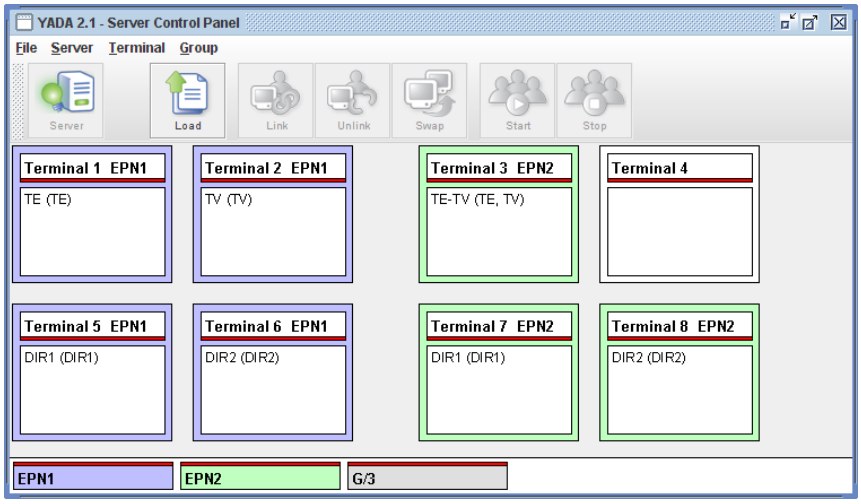


Note that a configuration file must be loaded and the server must be started for the terminal application to start properly. Also note that the visual communication terminal, shown above, is not started until the terminal is linked in the control panel and the assigned group is started. Continue to the next chapter for more information on how to do this.

5 Operating the control panel

The server control panel shows an overview of all terminals and groups defined in the configuration xml file. The groups are listed in the bottom with group name, color and a started indicator. Each terminal is visualized as a box containing a name field, online indicator and a role field. An unlinked terminal is uncolored and a linked terminal has the same color as the group it belongs to. The name of the group is appended to the terminal name in the upper text field. The lower text area hold a list of players and a list of roles in parantheses.

There is a menu bar in the top containing the items **File**, **Server**, **Terminal** and **Group** and a toolbar with icons. The icons become clickable when the action is allowed. The terminals can be selected by clicking them and deselected by clicking on an empty desktop area.



5.1 File menu

The file menu is used to load configurations and exercises and to exit the control panel. When exiting the contol panel the server will close and all terminals will be detached from the server and has to be restarted manually.

The **Load Configuration** selection opens up a file dialog to choose the configuration file to load.

The **Load Exercise** selection is used to load a predefined excerise file linking terminals with players and groups. Refer to chapter 6 for details about the exercise files.

5.2 Server menu

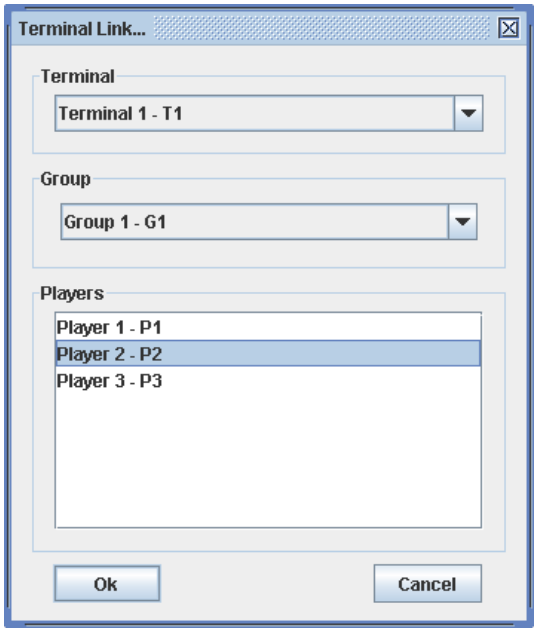
The server menu is used to control the communication server. The server must be started to use most of the functions in the control panel. This menu contains a **Start** and a **Stop** selection. When stopping the server all terminals will be detached and unlinked and has to be restarted manually.

5.3 Terminal menu

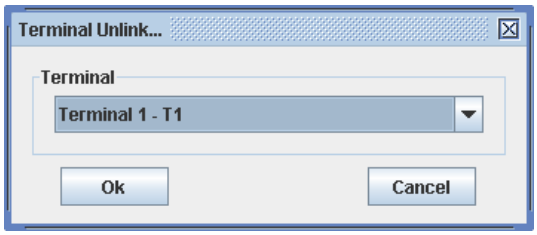
From the terminal menu players and groups can be linked, unlinked and swapped. All selections require a loaded configuration and a started server.

The **Link** selection opens up a link dialog. A terminal, a group and one or more players are selected in the dialog to link the terminal.

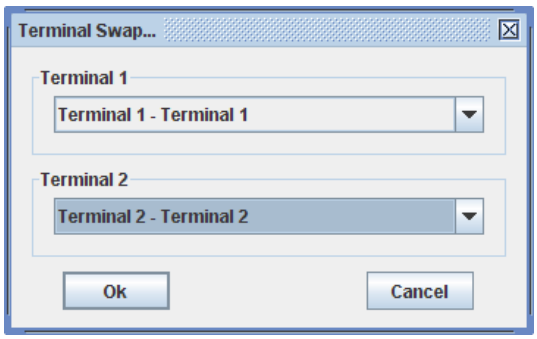
If two or more players are selected they are automatically merged to be able to utilize all channels and roles. The selected player is linked to the terminal selected in the terminal box and belongs to the group selected in the group box.



The **Unlink** selection opens a dialog to unlink a previously linked terminal. The terminal to unlink is selected in the dialog. The **Unlink All** selection unlinks all terminals.

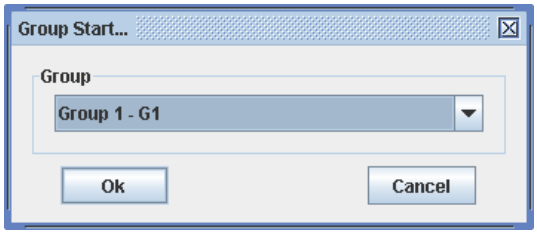


The **Swap** selection opens up a dialog to move an already linked terminal to an unlinked terminal or to swap two linked terminals. One terminal is selected in the upper box and another terminal is selected in the lower box.



5.4 Group menu

This menu is used to control the groups. Each group can be started, stopped and paused individually. All selections require a loaded configuration and a started server. A group must be started for the terminal application to start. The menu contains a **Start**, **Pause** and **Stop** selection. All selections open a dialog to choose which group to perform the action on.



6 Creating a predefined exercise

An exercise can be set up manually based on the definitions in the configuration file using the terminal menu. This is a quite time consuming task and not recommended. Preferrably a predefined exercise file is loaded and the features in the terminal menu are used only to modify the existing exercise.

Derive the new exercise file from the template found in **data/exercise/exercise-template.xml**. One simple and another more advanced example relating to the example configurations are found in the same folder. All id attributes must have values between 1 and 4000.

```
<Exercise>
    <TerminalLink id="1" groupid="1">
        <PlayerRef id="1"/>
        ...
    </TerminalLink>
    ...
</Exercise>
```

If the id attribute on the TerminalLink node is omitted the players will automatically be placed on unlinked terminals. If the groupid attribute is omitted, the supervisor will be asked to which group the players with no group id specified should belong.