SW6000 User Manual

CAA Name Sign Template

User guide for Shure SW6000 Conference Management Software

Version: 9.2 (2021)

Table of Contents

1 Introduction

This document describes the format of name sign template markup language as used in SW6000 in ‘CAA|Setup|Configuration|Name sign templates’.

The name sign template format used in SW6000 is in fact the render format directly understood by the rendering engine in the name sign hardware. In SW6000, though, the template is split into a “Front” and “Rear” definition for easier handling, and then merged into one before sending to a name sign.

So a template in SW6000 is in its essence a name sign render JSON file for rendering the front and rear displays on a name sign.

On top of the basic template definition SW6000 has a markup language which is embedded into the templates and which is evaluated by SW6000 when it renders the template for a specific name sign.

In this way e.g. participant information can be merged into the predefined name sign template before it is sent to a name sign.

What is covered in this document is the name sign template markup language defined for the CUI rendering of name sign content.

2 Template File

This chapter describes all tags used in a template file.

2.1 Object tags

The purpose of this chapter is to describe display object tags

Tag	Type	Description	Example
"BackgroundColor"	Optional number	Color to use for global background. Default color: 0xFFFFFFFF	"BackgroundColor" : 0xFFFFFFFF "Content" : […..
"ForegroundColor"	Optional number	Color to use for global text rendering. Default color: 0xFF000000	"ForegroundColor" : 0xNa000000, "Content" : […..
"Rotation"	Optional number	Number describing the number of degrees the image is to be rotated. Current support degrees are 0 and 180. Default rotation is 0.	"Rotation" : 180, "Content" : […..
"Content"	Optional array	This array contains a description of the content to be rendered. If no content is available the screen is cleared. Each item in the array shall include a unique 'ID'. **Note:** Conditional statement can be used in the content as well. Refer to: '4 Conditional statements'	"Content" : [{ "ID" : 1, "Type": "TextBox", …….. }, { "ID" : 2, "Type": "TextBox", …….. }]
"Invert"	Optional Boolean	Flag to signal content is to be inverted. Default value "false"	"Invert" : true, "Content" : […..
”Language”	Optional text	Language code (Locale Id). Specifies the SW6000 language, from where the data is collected. If no language code is specified or data is not available in the specified language, the 'System language' is used. If the resulting language code in use, is not included in the 'MXC Global languages' then the sign will use English (US), language code 1033.	”Language” : “1034”, "Content" : [
“ID”	Number	Identifies an object in the content array	

2.2 TextBox

This section describe objects which can be within a text box.

Note: A text box do not show any borders.

Tag	Type	Description	Example
"Type"	Mandatory string	Value must be "TextBox".	"Content" : [{ "ID" : 1, "Type": "TextBox", ... }]
"Font"	Optional string	String containing name of font to use for text box. Default font is based on language configuration, set by "Language" tag. Overrides the default font.	"Content" : [{ "ID" : 1, "Type": "TextBox", "Font": "Kanit", ... }]
"Size"	Optional number	Size of the font to use. Default size is set to 72.	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 130, ... }]
"X" "Y"	Optional number	X / Y start position of text box. Default X / Y position is set to 0. Position is calculated from upper left corner of the display	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 130, "X": 0, "Y": 0, ... }]
"Width" "Height"	Optional number	Width / height of text box Default width is set to panel max width (1904). Default height is set to panel max height (464).	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 130, "X": 0, "Y": 0, "Width" : 1904, "Height" : 464, ... }]

Tag	Type	Description	Example
"AlignH" "AlignV"	Optional string	Horizontal / vertical alignment of text within the text box. If not set, "Center" value is used. Valid horizontal values: "Left" "Right" "Center" Valid vertical values: "Top" "Bottom" "Center"	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 130, "X": 0, "Y": 0, "Width" : 1904, "Height" : 464, "AlignH": "Center", "AlignV": "Center", ... }]
"Text"	Mandatory string	UTF8 Encoded text. The string can include a text and/or a 'command' from SW6000 to insert a value. For 'commands' refer to '3.1 List of commands	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 130, "X": 0, "Y": 0, "Width" : 1904, "Height" : 464, "AlignH": "Center", "AlignV": "Center", "Text": "@SeatA(ParticipantName)" }]
"BackgroundColor"	Optional number	Color used to render background. Overrides global "BackgroundColor"	"ID" : 1, "Type": "TextBox", "BackgroundColor" : 0xFFFFFFFF
"ForegroundColor"	Optional number	Color to render text with. Overrides global "ForegroundColor" if used.	"ID" : 1, "Type": "TextBox", "ForegroundColor" : 0xFF000000,
"Language"	Optional text	Language code (Locale Id). Specifies the SW6000 language, from where the data is collected. If a language code is not specified, the 'Language', which may be specified in 'Object tag' is used, else the 'System language' is used. If a language code is specified, but data is not available in that language, the 'System language' is used. If the resulting language code in use, is not included in the 'MXC Global languages' then the sign will use English (US), language code 1033.	"ID" : 1, "Type": "TextBox", "Language" : "1033",

Tag	Type	Description	Example
		The language code ‘default’ is also valid.	"ID" : 2, "Type": "TextBox", "Language" : “default”,
		As the data to show is taken from the language data in SW6000, data in different languages, if available, can be shown simultaneously in the sign. In the example, the participant name shown on the top is the English name and in the bottom the Arabic name	"Content" : [{ "ID" : 1, "Type": "TextBox", "Size": 100, "Language" : "1033", "X": 0, "Y": 0, "Width" : 1904, "Height" : 232, "AlignH": "Center", "AlignV": "Center", "Text": "@SeatA(ParticipantName)" }, { "ID" : 2, "Type": "TextBox", "Size": 100, "Language" : "1025", "X": 0, "Y": 232, "Width" : 1904, "Height" : 232, "AlignH": "Center", "AlignV": "Center", "Text": "@SeatA(ParticipantName)" }]
"Style"	Optional string	Specifies style of font. Valid values are: "Regular" "Italic" "Bold" "BoldItalic" Default is "Regular". The font specified by "Font" must define a valid font for the style, or it will fall back to "Regular"	"ID" : 1, "Type": "TextBox", "Style" : Bold,

2.3 Comments

The template format supports inserting comments using the following format:

Hex	Description	Example
//	Used to insert comments or 'disable' a line. Active until a line break.	"Content" : [{ "ID" : 1, "Type": "TextBox", // "Font": "Kanit", // This is a Thai font .... }]
/* ..... */	Used to insert comments or 'disable' all in-between.	"Content" : [/*{ "ID" : 1, "Type": "TextBox", ... },*/ { "ID" : 2, "Type": "TextBox", ... }]

Important: Although the current template format supports inserting comments, this is not supported in standard json formats. This feature may be discontinued in future versions, and it is recommended to remove comments in the templates in use.

3 Template Command Format

The name sign template markup format rules:

- All commands starts with an @-sign
- Command arguments are passed in a pair of parentheses directly after the command (i.e. no white spaces between command and arguments)
- If there are no arguments to the command, the argument parentheses can be omitted.
- If a @-character is required in the template, it must be typed in as “@@” – the template renderer will render that sequence as a single @-character in the output.
- Commands and arguments cannot span multiple lines. Both must be on the same line in the template.

3.1 List of commands

The markup language contains the following commands used for applying content.

Command	Description
@SeatA	Insert value from paired sets A
@SeatB	Insert value from paired seat B
@Meeting	Insert value from the started meeting
@Sign	Insert value from name sign configuration
@Booth	Insert value from booth configuration

The following sections describes the commands in more detail.

3.2 @SeatA and @SeatB commands

The @SeatA/B commands are used for accessing information about the seat paired with the name sign as either A or B seat.

The command takes one argument specifying which information to fetch from the seat

Argument	Data inserted when rendering	Example
SeatNumber	The seat number of the paired seat	"Text": “Seat @SeatA(SeatNumber)” “Seat 5” – providing the name sign has seat 5 as paired seat A
ParticipantName	The full name of the participant of the paired seat	"Text": “@SeatB(ParticipantName)” “Peter Fessler” – providing the participant Peter Fessler is either logged in at, or assigned to, the seat paired as seat B on the name sign
ParticipantFirstName	The first name of the participant of the paired seat	
ParticipantLastName	The last name of the participant of the paired seat	
ParticipantShowName	The ‘show name’ of the participant of the paired seat. Show name is configured in ‘CAA|Setup|Meeting role’	.
ParticipantTitle	The title of the participant of the paired seat	
ParticipantCustom1 ParticipantCustom2 ParticipantCustom3 ParticipantCustom4	The “User Table 1-4” value of the participant of the paired seat	
GroupName	The group name of the participant of the paired seat	
GroupAbbreviation	The abbreviated group name of the participant of the paired seat	
Message	A message send to a participant	"Text": "@SeatA(Message)"

Most values in the table above refer to “the participant of the paired seat”. This is the participant intended to be displayed on the name sign. First and foremost, this is the participant which is logged in at the seat. If no participant is logged in, and there is a seat assignment in the current meeting, the participant assigned to the seat will become “the participant of the paired seat”, providing the participant is not currently logged in at another seat.

If there is no participant for the paired seat, the participant-derived values will render as an empty string.

All text values are fetched in the system default language for all name signs unless a font or a language code is specified.

3.3 @Sign command

The @Sign command is used to access information about the name sign itself.

The command takes one argument specifying which information to fetch from the seat

Argument	Data inserted when rendering	Example
SerialNumber	The serial number of the name sign	"Text": “SerialNo: @Sign(SerialNumber)” “SerialNo: 164.214.045” – providing the serial number of the name sign is 164.214.045
SeatA	Seat number of the seat paired as A seat on the name sign	"Text": "Seat @Sign(SeatA)"
SeatB	Seat number of the seat paired as B seat on the name sign	"Text": "Seat @Sign(SeatB)"

3.4 @Meeting command

The @Meeting command is used to access information about the active meeting.

The command takes one argument specifying which information to fetch from the seat

Argument	Data inserted when rendering	Example
MeetingName	The name of the active meeting	"Text": "@Meeting(MeetingName)"
ActiveSubject	The title of the active subject	"Text": "@Meeting(Active subject)"

3.5 @Booth command

The @Booth command is used to access information about interpreter booth.

The command takes one argument specifying which information to fetch from the booth.

Argument	Data inserted when rendering	Example
BoothLanguage	The A language associated with a booth	"Text": "@Booth(BoothNumber) - @Booth(BoothLanguage) (@Booth(BoothLanguageAbbreviation))"
BoothLanguageAbbreviation	Abbreviation for the A language associated with a booth	
BoothNumber	The booth number	
BoothLanguageChannel	The number of the A channel associated with a booth	

4 Conditional Statements

The conditional statement can be achieved in two ways:

- Conditional content in Text properties using @If
- Conditional JSON sections using “If” objects

The following sections describes the conditional statements in details.

4.1 Conditional content in Text properties using @If

The name sign conditional commands used as ‘Text’ arguments:

- All commands starts with an @-sign
- Command arguments are passed in a pair of parentheses directly after the command (i.e. no white spaces between command and arguments)
- If there are no arguments to the command, the argument parentheses can be omitted.
- If the first character after a command or command argument end parenthesis is a space, it is trimmed out when rendering a template. This allows the output of a markup command to line up with literal content in the template. (e.g. the template snippet “’@If(something) Yes@Else No” would evaluate to “Yes” or “No” when rendering – without skipping a space after a command it would yield “Yes” or “ No”, where No has a leading space, or would require empty parentheses after @Else to be able to put “No” right after the @Else to avoid the space)
- If a @-character is required in the template, it must be typed in as “@@” – the template renderer will render that sequence as a single @-character in the output.
- Commands and arguments cannot span multiple lines. Both must be on the same line in the template.

4.1.1 List of commands

The markup language contains the following commands

Command	Description
@If	Open a conditional section in the Text content
@Else	Open the alternative section of a conditional section in the Text content
@EndIf	Close a conditional section in the Text content

4.1.2 @If...@Else...@EndIf commands

The @If, @Else and @EndIf commands exists for conditional inclusion of sections in Text property values.

Everything between an @If and its associated @Else command is rendered to the name sign by the CUI only if the condition argument to the @If command is true. Otherwise, the section between the @Else and its associated @EndIf command is rendered to the name sign.

The @Else command can be excluded, in which case the section between the @If and @EndIf commands is rendered if the condition argument of the @If command is true.

@If..@Else..@EndIf constructs can be nested for complex conditional template content.

The @If command takes one of the following conditions as argument.

Argument	Condition is true when
SeatAHasParticipant	The seat paired as name sign seat A has a participant assigned to it (i.e. a participant is to be displayed on the name sign)
SeatBHasParticipant	The seat paired as name sign seat B has a participant assigned to it (i.e. a participant is to be displayed on the name sign)

The @Else and @EndIf commands take no arguments.

4.1.3 Example using @If

The following examples includes some JSON formatting, since the @If...@Else...@EndIf command syntax is designed to include or exclude chunks of text data in the template.

Template snippet	Output
"Text": "@If(SeatAHasParticipant) Participant A is here @If(SeatBHasParticipant) with participant B @EndIf @Else @If(SeatBHasParticipant) Participant B is here alone @Else No one is here @EndIf @EndIf",	***No one is here*** If no participant is assigned to either paired seat A or B. ***Participant A is here*** If a participant is assigned to paired seat A and no one is assigned to paired seat B. ***Participant B is here alone*** If a participant is assigned to seat B and no one is assigned to seat A. ***Participant A is here with participant B*** If both seat A and B has assigned participants.

4.1.4 Notes on white spaces

To limit the amount of data sent over the DCS LAN, the CUI will trim down the JSON of name sign templates. To save on processing this is done before the template is pre-parsed into literal text and markup commands.

The trimming down of the JSON involves removing all non-quoted spaces and line breaks. This will affect how the markup language is parsed after the trimming. If, for instance, you have the construct “@EndIf This is my text” it will be trimmed down to ”@EndIfThisIsMyText”, which will fail parsing, since there is no markup command named @EndIfThisIsMyText.

A workaround for such a situation would be to include the optional parantheses on @EndIf. This would result in the trimmed down “@EndIf()ThisIsMyText”, which is parsable.

4.2 Conditional JSON sections using "If" objects

Command	Description
If	Open a conditional section in the template
Condition	Specified the condition
Then	Open the conditional section in the template
Else	Open an optional conditional section in the template

The condition command takes the same conditions as argument as the @If command.

Condition	Condition is true when
SeatAHasParticipant	The seat paired as name sign seat A has a participant assigned to it (i.e. a participant is to be displayed on the name sign)
SeatBHasParticipant	The seat paired as name sign seat B has a participant assigned to it (i.e. a participant is to be displayed on the name sign)

4.2.1 Example with 'If, then, else'

```
"Content" :
[
  {
   "Type": "If",
   "Condition": "SeatAHasParticipant",
   "Then":
   [
     {
     "ID"  : 1,
     "Type": "TextBox",
     "Size": 90,
     "X": 0,
     "Y": 0,
     "Width" : 1904,
     "Height" : 230,
     "AlignH": "Left",
     "AlignV": "Center",
     "Text": "Seat A has participant"
     }
   ],
   "Else":
   [
     {
     "ID"  : 2,
     "Type": "TextBox",
     "Size": 90,
     "X": 0,
     "Y": 0,
     "Width" : 150,
     "Height" : 230,
     "AlignH": "Left",
     "AlignV": "Center",
     "Text": "Seat A has no participant"
     }
   ]
  }
                              ]
```

4.3 Comparison between "Type" : "If" and @If

The following template examples shown the two ways of using 'If' to constructs the display a "<" and ">" direction indicator if somebody is logged in at the A and B seat respectively, indicating the seating position of the displayed participant name.

Achieved with "Type": "If"

```
"Content" :
[
  {
   "Type": "If",
   "Condition": "SeatBHasParticipant",
   "Then":
   [
     {
      "ID"  : 1,
      "Type": "TextBox",
      "Size": 90,
      "X": 0,
      "Y": 0,
      "Width" : 1904,
      "Height" : 230,
      "AlignH": "Left",
      "AlignV": "Center",
      "Text": "< @SeatB(ParticipantName)"
     }
   ]

  },
  {
   "Type": "If",
   "Condition": "SeatAHasParticipant",
   "Then":
   [
     {
      "ID"  : 4,
      "Type": "TextBox",
      "Size": 90,
      "Font" : "Kanit",
      "X": 0,
      "Y": 230,
      "Width" : 1904,
      "Height" : 230,
      "AlignH": "Right",
      "AlignV": "Center",
      "Text": "@SeatA(ParticipantName) >"
     }
   ]

  }
]
```

Achieved with @If … @Endif

```
"Content" :
[
  {
      "ID"  : 1,
      "Type": "TextBox",
      "Size": 90,
      "X": 0,
      "Y": 0,
      "Width" : 1904,
      "Height" : 230,
      "AlignH": "Left",
      "AlignV": "Center",
     "Text": "@If(SeatBHasParticipant)< @SeatB(ParticipantName) @EndIf",
  },
  {
      "ID"  : 4,
      "Type": "TextBox",
      "Size": 90,
      "Font" : "Kanit",
      "X": 0,
      "Y": 230,
      "Width" : 1904,
      "Height" : 230,
      "AlignH": "Right",
      "AlignV": "Center",
      "Text": "@If(SeatAHasParticipant) @SeatA(ParticipantName) >@EndIf",
   }
]
```

5 Fonts

The font sizes supported are TTF fonts and all pt sizes are valid.

If an invalid font is specified in a template, the “Roboto Condensed Regular” font will be used.

5.1 Supported fonts including style availability

The next table shows the fonts available in the Namesign.

Font	Regular	Italic	Bold	Bold Italic
Roboto	X	x	x	x
RobotoCondensed	X	x	x	x
NotoSansThai	X		x	
Kanit	X		x	
NotoSansHebrew	X		x	
NotoSansCJK	X			
NotoSansArabic	X		x	
NotoNaskhArabic	X		x	

5.2 MXC global languages

The table shows the default font selection for the MXC global languages. Each language is defined with a language code (locale id).

Language	Language code (Locale Id)	Supported by font
Arabic	1025	NotoNaskhArabic
Basque	1069	RobotoCondensed
Chinese Simple	2052	NotoSansCJK
Chinese Traditional	1028	NotoSansCJK
Catalan	1027	RobotoCondensed
Dutch	1043	RobotoCondensed
English	1033	RobotoCondensed
French	1036	RobotoCondensed
German	1031	RobotoCondensed
Indonesian	1057	RobotoCondensed
Italian	1040	RobotoCondensed
Japanese	1041	NotoSansCJK
Korean	1042	NotoSansCJK
Lithuanian	1063	RobotoCondensed
Portuguese	1046	RobotoCondensed
Russian	1049	RobotoCondensed
Spanish	3082	RobotoCondensed
Thai	1054	Kanit
Turkish	1055	RobotoCondensed

5.3 Fonts in use with SW6000

When the MXCSIGN is used with SW6000 and the language(s) used in SW6000 is not supported in the MXC global languages, the MXCSIGN will use the Language code (Locale Id) 1033 (RobotoCondensed font).

It is therefore not needed to change the 'Language' in any of the default templates in SW6000 if the used language is a language supported in the RobotoCondensed font.

The Roboto and RobotoCondensed fonts support Latin, Greek and Cyrillic script and the following characters are supported:

" # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z {|} ~ ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ Ā ā Ă ă Ą ą Ć ć Ĉ ĉ Ċ ċ Č č Ď ď Đ đ Ē ē Ĕ ĕ Ė ė Ę ę Ě ě Ĝ ĝ Ğ ğ Ġ ġ Ģ ģ Ĥ ĥ Ħ ħ Ĩ ĩ Ī ī Ĭ ĭ Į į İ ı IJ ij Ĵ ĵ Ķ ķ ĸ Ĺ ĺ Ļ ļ Ľ ľ Ŀ ŀ Ł ł Ń ń Ņ ņ Ň ň ŉ Ŋ ŋ Ō ō Ŏ ŏ Ő ő Œ œ Ŕ ŕ Ŗ ŗ Ř ř Ś ś Ŝ ŝ Ş ş Š š Ţ ţ Ť ť Ŧ ŧ Ũ ũ Ū ū Ŭ ŭ Ů ů Ű ű Ų ų Ŵ ŵ Ŷ ŷ Ÿ Ź ź Ż ż Ž ž ſ ƒ Ơ ơ Ư ư ǰ Ǻ ǻ Ǽ ǽ Ǿ ǿ Ș ș Ț ț ˆ ˇ ˘ ˙ ˚ ˛ ˜ ˝ ̀ ́ ̃ ̉ ̣ ̈ ̓ Ά · Έ Ή Ί Ό Ύ Ώ ΐ Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω Ϊ Ϋ ά έ ή ί ΰ α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ϊ ϋ ό ύ ώ ϑ ϒ π Ѐ Ё Ђ Ѓ Є Ѕ І Ї Ј Љ Њ Ћ Ќ Ѝ Ў Џ А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я а б в г д е ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я ѐ ё ђ ѓ є ѕ і ї ј љ њ ћ ќ ѝ ў џ Ѡ ѡ Ѣ ѣ Ѥ ѥ Ѧ ѧ Ѩ ѩ Ѫ ѫ Ѭ ѭ Ѯ ѯ Ѱ ѱ Ѳ ѳ Ѵ ѵ Ѷ ѷ Ѹ ѹ Ѻ ѻ Ѽ ѽ Ѿ ѿ Ҁ ҁ ҂ ҃ ҄ ҅ ҆ ҇ Ҋ ҋ Ҍ ҍ Ҏ ҏ Ґ ґ Ғ ғ Ҕ ҕ Җ ж ҙ ҙ Қ қ Ҝ ҝ Ҟ ҟ Ҡ ҡ Ң ң Ҥ ҥ Ҧ ҧ Ҩ ҩ Ҫ ҫ Ҭ ҭ Ү ү Ұ ұ Ҳ ҳ Ҵ ҵ Ҷ ҷ Ҹ ҹ Һ һ Ҽ ҽ Ҿ ҿ Ӏ Ӂ ӂ Ӄ ӄ Ӆ ӆ Ӈ ӈ Ӊ ӊ Ӌ ӌ Ӎ ӎ Ӏ Ӑ ӑ Ӓ ӓ Ӕ ӕ Ӗ ӗ Ә ә Ӛ ӛ Ӝ ӝ Ӟ ӟ Ӡ ӡ Ӣ ӣ Ӥ ӥ Ӧ ӧ Ө ө Ӫ ӫ Ӭ ӭ Ӯ ӯ Ӱ ӱ Ӳ ӳ Ӵ ӵ Ӷ ӷ Ӹ ӹ Ӻ ӻ Ӽ ӽ Ӿ ӿ Ԁ ԁ Ԃ ԃ Ԅ ԅ Ԇ ԇ Ԉ ԉ Ԋ ԋ Ԍ ԍ Ԏ ԏ Ԑ ԑ Ԓ ԓ Ԕ ԕ Ԗ ԗ Ԙ ԙ Ԛ ԛ Ԝ ԝ Ԟ ԟ Ԡ ԡ Ԣ ԣ Ԥ ԥ Ԧ ԧ Ԩ ԩ Ԫ ԫ Ԭ ԭ Ԯ ԯ ՠ ա բ գ դ ե զ է ը թ ժ ի լ խ ծ կ հ ձ ղ ճ մ յ ն շ ո չ պ ջ ռ ս վ տ ր ց ւ փ ք օ ֆ ։ ֊ ־ ׀ ׃ ׆ א ב ג ד ה ו ז ח ט י ך כ ל ם מ ן נ ס ע ף פ ץ צ ק ר ש ת ׯ װ ױ ׳ ״ ׿ ؀ ؁ ؂ ؃ ؄ ؅ ؆ ؇ ؈ ؉ ؊ ؋ ، ؍ ؎ ؏ ؐ ؑ ؒ ؓ ؔ ؕ ؖ ؗ ؘ ؙ ؚ ؛ ؜ ؝ ؞ ؟ ؠ ء آ أ ؤ إ ئ ا ب ة ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ـ ف ق ك ل م ن ه و ى ي ً ٌ ٍ َ ُ ِ ّ ْ ٓ ٔ ٕ ٖ ٗ ٘ ٙ ٚ ٛ ٜ ٝ ٞ ٟ ٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ ٪ ٫ ٬ ٭ ٮ ٯ ٰ ٱ ٲ ٳ ٴ ٵ ٶ ٷ ٸ ٹ ٺ ٻ ټ ٽ پ ٿ ڀ ځ ڂ ڃ ڄ څ چ ڇ ڈ ډ ڊ ڋ ڌ ڍ ڎ ڏ ڐ ڑ ڒ ړ ڔ ڕ ږ ڗ ژ ڙ ښ ڛ ڜ ڝ ڞ ڟ ڠ ڡ ڢ ڣ ڤ ڥ ڦ ڧ ڨ ک ڪ ګ ڬ ڭ ڮ گ ڰ ڱ ڲ ڳ ڴ ڵ ڶ ڷ ڸ ڹ ں ڻ ڼ ڽ ھ ڿ ۀ ہ ۂ ۃ ۄ ۅ ۆ ۇ ۈ ۉ ۊ ۋ ی ۍ ێ ۏ ې ۑ ے ۓ ە ۖ ۗ ۘ ۙ ۚ ۛ ۜ ۝ ۞ ۟ ۠ ۡ ۢ ۣ ۤ ۥ ۦ ۧ ۨ ۩ ۪ ۫ ۬ ۭ ۮ ۯ ۰ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ ۺ ۻ ۼ ۽ ۾ ۿ ‒ – — ― ‗ ‘ ’ ‚ ‛ “ ” „ † ‡ • ‣ ‥ … ‰ ′ ″ ‹ › ‼ ⁄ ⁴ ⁿ ₣ £ ₧ ₫ € ℅ ℓ № ™ ℮ ⅛ ⅜ ⅝ ⅞ ∂ ∏ ∑ − √ ∞ ∫ ≠ ≤ ≥ ◊ ﷼ fi fl ffi ffl

5.3.1 Supported languages in Roboto and RobotoCondensed

The following languages are fully or partly supported in Roboto and RobotoCondensed fonts:

Afrikaans	English	Irish	Rhaeto-Romanic
Albanian	Esperanto	Italian	Russian
Bashkir	Estonian	Kayah li	Scots
Basque	Faroese	Kazakh	Scottish Gaelic
Belarusian	Finnish	Kurdish	Southern Sami
Breton	French	Latin	Spanish
Bulgarian	Galician	Leonese	Swahili
Catalan	German	Luxembourgish	Swedish
Chinese pinyin	Greek	Malay	Tagalog
Corsican	Hungarian	Manx	Tatar
Cyrillic	Icelandic	Norwegian	Turkish
Czech	Indonesian	Occitan	Ukrainian
Danish	Internat. phonetic	Polish	Walloon
Dutch	Irish	Portuguese	Welsh

5.4 Hebrew language

Hebrew is not supported in the MXC global language, but Hebrew is supported in the MXCSIGN:

Language	Language code (Locale Id)	Supported by font
Hebrew	1037	NotoSansHebrew

To use Hebrew, the Hebrew language has to be specified in the Object tags or in a TextBox in the template.

6 Grey Colors

6.1 Color format

The name sign support 16 levels of grey colors. The color format used is 0xAARRGGBB.

AA – 8bit Alpha channel.
RR – 8bit Red channel.
GG – 8bit Green channel.
BB – 8bit Blue channel

Any color selected that do not support native colors will be converted to nearest grey and dithered.

The JSON only accepts the Hex [A,R,G,B] and Decimal formats.

6.2 Native colors supported

Hex [A,R,G,B]	Decimal	List
0xFF000000	4278190080	[255,0,0,0]
0xFF111111	4279308561	[255,17,17,17]
0xFF222222	4280427042	[255,34,34,34]
0xFF333333	4281545523	[255,51,51,51]
0xFF444444	4282664004	[255,68,68,68]
0xFF555555	4283782485	[255,85,85,85]
0xFF666666	4284900966	[255,102,102,102]
0xFF777777	4286019447	[255,119,119,119]
0xFF888888	4287137928	[255,136,136,136]
0xFF999999	4288256409	[255,153,153,153]
0xFFAAAAAA	4289374890	[255,170,170,170]
0xFFBBBBBB	4290493371	[255,187,187,187]
0xFFCCCCCC	4291611852	[255,204,204,204]
0xFFDDDDDD	4292730333	[255,221,221,221]
0xFFEEEEEE	4293848814	[255,238,238,238]

7 Name Sign Specifications

The MXCSIGN has the following specifications:

Image Size	1904 x 464 pixels
Screen Dimensions	380mm x 100 mm
Thickness	9mm
Width x Height	402mm x 113mm

The positions and size of an item are specified in pixels. The position of an item is counted from upper left corner of the display.

Font sizes are specified in pt.

www.shure.com

United States, Canada, Latin America, Caribbean:
Shure Incorporated
5800 West Touhy Avenue
Niles, IL 60714-4608
USA

Phone: +1 847 600 2000
Fax: +1 847 600 1212 (USA)
Fax: +1 847 600 6446
Email: info@shure.com

Europe, Middle East, Africa:

Shure Europe Gmbh
Jakob-Dieffenbacher-Str. 12
75031 Eppingen
Germany

Phone: +49 (0) 7262-9249-100
Fax: +49 (0) 7262-9249-114
Email: info@shure.de

Asia, Pacific:

Shure Asia Limited
22/F, 625 King's Road
North Point, Island East,
Hong Kong

Phone: (+852) 2893-4290
Fax: (+852) 2893-4055
Email: info@shure.com.hk