Elektrobit

EB GUIDE Studio

Tips for smart modeling

Version 6.9 or later

Elektrobit Automotive GmbH
Am Wolfsmantel 46
D-91058 Erlangen
GERMANY

Phone: +49 9131 7701-0
Fax: +49 9131 7701-6333
http://www.elektrobit.com

Legal notice

Table of Contents

1. Begin here

This document provides tips for projects with EB GUIDE Studio and EB GUIDE GTF. These tips are based on experiences with EB GUIDE projects at EB. The provided information spans different topics. Consider that your own modeling or coding guidelines may influence how you interpret these tips.

If this document does not cover a topic that you are interested in, please contact the support.

EB GUIDE support is available in the following ways:

- For the community edition, find comprehensive information in our articles, blogs, and user documentation.
- For the enterprise edition, contact us according to your support contract.

When you look for support, prepare the version number of your EB GUIDE Studio installation. To find the version number, open EB GUIDE Studio, go to the project center, and click **Help**. The version number is located in the lower right corner of the dialog.

NOTE

Change without notice

This document is being updated in irregular intervals and is subject to change without notice.

2. Performance

WARNING **Performance-related issues**

Adhere to the tips below concerning performance. Take hardware-related limitations into account. Otherwise your EB GUIDE project can have performance-related issues.

1. Avoid complex views.
   - Prefer a more complex state machine with several simple views over a simple state machine with complex views.
   - Limit the number of widgets in the widget template tree. Large widget template trees have a negative impact on performance. Templates are added to the generated model data as often as there are instances of a template. Adding many widgets creates a lot of overhead. Use solutions with fewer widgets.
   - Invisible widgets or sub-trees also increase the load when the widget tree is created.
   - Avoid views where large numbers of widgets are set to invisible during run-time. Invisible widgets are still processed to keep the EB GUIDE model consistent, only the rendering is not executed. A better solution is to use separate views. If you want elements to move in and out of the view, use dynamic state machines.
2. Reduce EB GUIDE Script execution.
   - Some widget properties should not have scripted values. This concerns properties such as `x`, `y`, `width`, `height`, `focused`, `pressed`, `touched`. These values are read by the renderer. Modeling them as EB GUIDE Script affects the performance. Instead create a separate EB GUIDE Script to set these properties.
   - When you model dynamic behavior with EB GUIDE Script and widget features, always be aware which properties may be read and updated by the built-in widget feature to avoid update cycles or conflicting updates. For example, layout changes require updates of multiple property values.
   - Prefer the default animation curves over script curves. Script curves require additional processing steps because they are defined in EB GUIDE Script.
   - For complex calculations, foreign functions are faster than EB GUIDE Script .
   - If foreign functions involve operations on resources (fonts, images) or have a large number of parameters, they are especially costly.
3. Avoid unstable states. Your EB GUIDE model should reach a stable state quickly to be able to work together with the EB GUIDE GTF update processing cycles. For more information, see the EB GUIDE GTF user documentation section "Update processing in EB GUIDE GTF".

Example 2.1.

Alternating images

This example shows what a recommended and what a not recommended implementation looks like.

You want to model alternating displaying of 10 different images.

- Not recommended: Modeling 10 image widgets.
- Recommended: Modeling one image widget with a reference to a list of 10 images.

Example 2.2.
Property values accessing list elements

This example shows what a recommended and what a not recommended implementation looks like.

You want to model a property that accesses an element from a list.

- Not recommended: Modeling the property with EB GUIDE Script.
- Recommended: Linking the property to a datapool item or a widget property of type list.

3. Animations

For more information, see EB GUIDE Studio user guide section "Animations".

4. Start and cancel animations in a consistent way throughout your EB GUIDE model. For example, use conditional scripts with consistent names.
5. Cancel a running animation before you restart it. Restarting it without canceling can cause lagging of the animation.
6. To disable animations, use the `enabled` property. Setting the `enabled` property to `false` cancels animations immediately.
7. Do not use the `visibility` property to disable animations. Disabling the `visibility` property does not disable the animation. The animation is still going to run.
8. Only the first and last step of an animation are guaranteed to be executed. Depending on system load, intermediate steps may be skipped.
9. To model animations during view transitions, use view transition animation (VTA). Using widget animation for this purpose requires additional steps to synchronize the animation with the view transition. VTA makes transition animations simpler.

4. EB GUIDE model changes check-in

These recommendations are for the revision control system (RCS), for example, SVN, Git, ClearCase. In order to have a functioning EB GUIDE model after committing your latest changes to the RCS, follow these steps.

Checking in EB GUIDE model changes

Prerequisite:

- An EB GUIDE model is opened in EB GUIDE Studio.

Step 1
Save the EB GUIDE model.

Step 2
Make an additional local backup copy of the EB GUIDE model.

Step 3
In the RCS check for modifications on the EB GUIDE model folder to check for non-versioned or missing files.

- If missing files are missing on purpose, delete them from the RCS.
- If new, non-versioned files are needed, add them to the RCS.

Step 4
In the RCS update the model folder.

Step 5

- If there are automatically merged files, review them. Somebody else could have changed something that is relevant to you.
- If there are deleted `.gdata` files or other EB GUIDE model elements, verify during review that references to them are removed throughout the change set.
- If there are conflicted files, work through the diffs and resolve the conflicts.
- If something is unclear, ask the person who committed the changes and resolve it together.
- If none of these points apply, continue with the next step.

Step 6
Reload the merged EB GUIDE model in EB GUIDE Studio.

Step 7
Validate the EB GUIDE model and resolve newly introduced errors.

Step 8
Start the simulation. If it runs properly, your EB GUIDE model is syntactically correct.

Step 9
If you changed something, save the EB GUIDE model.

Step 10
In the RCS commit the EB GUIDE model folder.

NOTE **`.ebguide` folder**

Exclude the `$GUIDE_PROJECT_PATH/<project name>/.ebguide` folder from the revision control system because it contains user-relevant settings.

5. EB GUIDE Script

For more information, see EB GUIDE Studio user guide sections "Scripting language EB GUIDE Script" and "Conditional script".

10. Use comments and indentations to document behavior.
11. Do not create update cycles. For example, a scripted condition that triggers an animation that sends an event after completion that triggers the same condition again.
12. Do not use animated properties as triggers.
13. Consider the scope of an EB GUIDE Script. Does it affect only a widget or also a datapool item? Usually widget properties only affect one widget or start an animation with scripted values with return type `void`. Datapool items usually control the configuration of the entire project, like the language. Model this kind of behavior with datapool items, as part of the state machine, or as part of a separate logical state machine.
14. If you want to run an EB GUIDE Script at every change of a language or skin, use a datapool item as a trigger that changes with the language or skin. EB GUIDE Script does not react on language or skin changes by default.
15. For complex scripts: Currently, EB GUIDE Script has no `switch case` or `break` functionality. Consider how many conditions have to be evaluated every time a script is executed. If possible:
    - Use `else` clauses.
    - To avoid repeating calculations, foreign function calls, or reading the same datapool value several times throughout a script, use local variables (`v:myScriptVar...`).
16. Conditional scripts with datapool items in the trigger are executed once per datapool context that is defined in the EB GUIDE model. For this reason, consider the following:
    - In a cascade of multiple if-else-if-else... check the most probable before the less probable cases.
    - Check whether the same logic could be moved to:
        - An internal transition, for example in the case of pop-up triggers
        - A state enter or exit action
        - A logic state machine, for example to keep track of model internal states, such as `phone call initiated` or `phone call ongoing`
        - A widget template, for example in the case of calculating the size of a widget
17. Use the default argument in conditional scripts (`v:arg0`) to enable or disable execution of your script contents on initialization. Within the condition `if(v:arg0)`:
    - Conditional scripts in the datapool are only executed once at start-up.
    - Conditional scripts in widgets are executed every time the widget is instantiated, for example every time you enter a view that contains the widget.

5.1. EB GUIDE Script formatting

18. Add spaces in front of and after operators such as `= / + .`
19. Add empty lines before and after code blocks. But only use one empty line. If `}` is directly followed by `}`, no empty line is necessary.
20. Indent lines after the beginning of a code block. Example:

```
Function(…)
{
   v:this.value=1
}
```

5.2. Foreign functions

21. Use built-in or user-defined foreign functions to encapsulate complicated business logic or longer calculations.
22. Every foreign function call within an EB GUIDE Script has a high initial overhead due to boxing/stacking of function parameters.
23. Reuse calculation results instead of computing the same result multiple times. For example, instead of calling a foreign function with same parameters repeatedly, store the result in a local variable and reuse it.

6. Naming

Naming conventions make it easier for your development team to work with the EB GUIDE model and with other team members. They prevent misunderstandings and common issues that come from a misinterpretation of names.

24. Use unique names.
25. Use speaking names. Names shall say what the object or property is used for.
26. Use underscores _ to separate prefixes and suffixes or for a meaningful separation of sub-strings.
27. Use internal capitalization (camel case). Internal capitalization means using a capital letter within a phrase, without any space or other separation of words.
    - Upper camel case: `UpperCase`
    - Lower camel case: `lowerCase`
28. Use namespaces to group content. Namespaces were introduced in EB GUIDE Studio 6.8. If you are using a version earlier than 6.8, use prefixes separated by underscore to group elements.
29. Do not use similar names that only differ in capital and small letters.
30. Do not use EB GUIDE Script keywords as names. When you use EB GUIDE Script distinguish clearly between model elements and code. For a list of all keywords, see EB GUIDE Studio user guide chapter "EB GUIDE Script keywords".
31. Do not use names of widget feature properties for user-defined properties. For a list of widget feature properties, see the EB GUIDE Studio user guide chapter "References".
32. Do not use white space. Replace white space with underscores.
33. Avoid naming elements in a way where they only differ in capitalization, hyphenation, spaces, or underscores, for example `PopupState` and `popupState`.
34. Stick to the following range of characters: `a-z, A-Z, 0-9` and underscore. Characters outside of this range may be replaced automatically during the export of an EB GUIDE model. It is easier to find elements in the exported files when no characters need to be replaced. For example, the following datapool item names result in the following `#define`.
    - `dp:var A` results in `#define LOCAL_DP_ID_VAR_A ((uint32_t) 0x00000000)`
    - `dp:var_A` results in `#define LOCAL_DP_ID_VAR_A1 ((uint32_t) 0x00000002)`
    - `dp:var-A` results in `#define LOCAL_DP_ID_VARA ((uint32_t) 0x00000001)`

6.1. Example naming guideline

This section provides an example naming guideline.

Animations

Start with prefix `a`, then underscore, and then the description. Examples:

- `a_Playback`
- `a_SpinningWheel`

Animation curves

Start with a description of what the animation curve does followed by the name of the target widget or datapool item which is animated. Examples:

- `incrementPlaybackTime`
- `rotateSpinningWheel`
- `enlargeSpinningWheel`

Datapool items

Start with a prefix and then underscore, and then the description. The prefix indicates the type.

Prefix	**Datapool item type**	**Example**
`c_`	Color	`c_White_EB`
`f_`	Font	`f_Narrow_50`
`s_ sl_`	String, String list - which needs language support	`s_Welcome`

Start the names of other datapool items with the context that they are used in. The names of other datapool items do not have any specific prefix. Examples:

- `Display_Width`
- `CD_IsPlaying`
- `CD_SongDuration`

Dynamic state machines

Start with prefix `dm`, then underscore, and then the description. Examples:

- `dm_Help`

- `dm_Warning`

Events

Start with a lower case letter. Use two words. The first word describes the action that the event triggers. The second word describes the context of the event. Separate the second word with an underscore. Write the second word with a capital letter. Examples:

- `goto_Main`
- `pause_CD`
- `increase_LogoSize`

Write parameters in lower camel case. Examples:

- `status`
- `isTimeOut`

States and views

Start with a prefix, then underscore, and then the description of the state. Possible prefixes are, for example, `vs` for view states or `cs` for compound states. Examples:

- `vs_StartUp`
- `cs_CD`
- `vs_Disambiguation`

Start names of views with prefix `v`, then underscore, and then the description. Examples:

- `v_StartUp`
- `v_CDPlayer`

Templates

Start with a prefix, then underscore, and then the description. Possible prefixes are `tb_` for buttons or `tv_` for views. Examples:

- `tb_Simple`
- `tv_Training`

User-defined properties

Start with a lower case letter and write the name in camel case. Start conditional scripts with an action. Examples:

- `buttonImage`
- `textX`
- `startAnimation`
- `setColor`

Widgets

Start with a prefix, then underscore, and then the description. Possible prefixes are `bg` for background, `l` or `t` for label, `bt` for button. Examples:

- `bg_Image`
- `l_Welcome`
- `bt_Play`

7. State machines

For more information, see EB GUIDE Studio user guide section "State machines and states".

35. Use dynamic state machines to model pop-ups such as warnings or error messages.
36. Use logical state machines to model complex behavior instead of using EB GUIDE Script or foreign functions.

Example 7.1.
Transitions

This example shows how a recommended and how a not recommended implementation looks like.

- Not recommended:

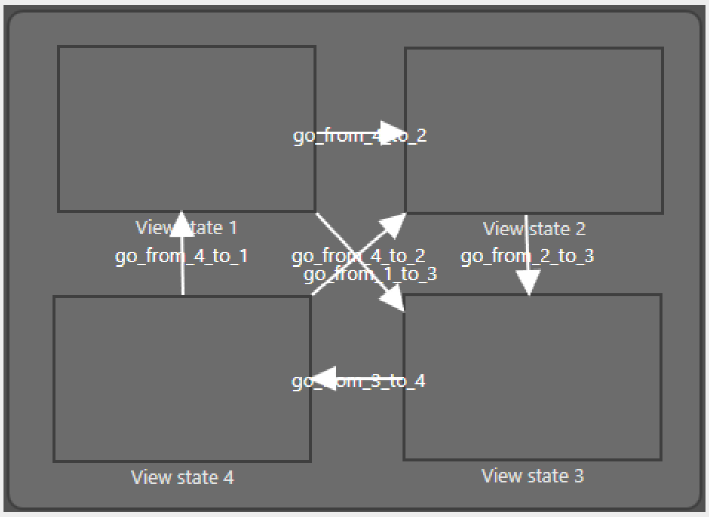

Figure 7.1. Not recommended transitions

- Recommended:

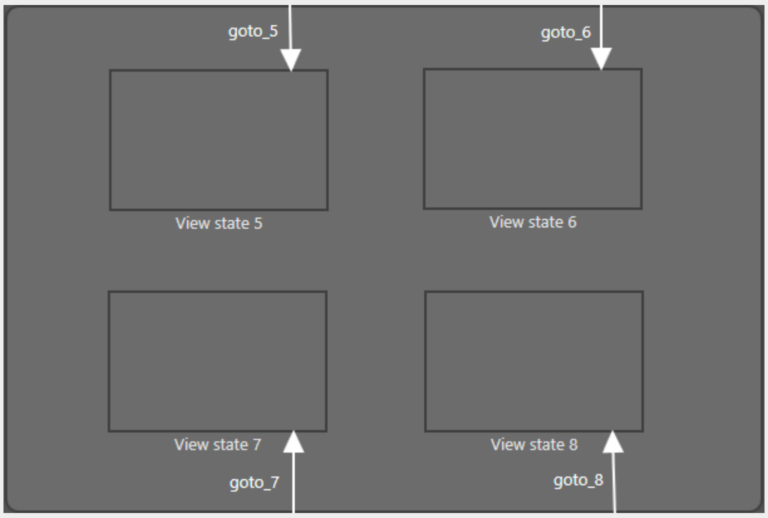

Figure 7.2. Recommended transitions

8. Templates

For more information, see EB GUIDE Studio user guide section "Widget templates".

37. If a property of a template is supposed to be set from the outside, add this property to the template interface.
38. If a property of a template is not supposed to be set from the outside, do not add this property to the template interface.
39. Make templates small and interchangeable to make the reuse easier. For example, if you have ten views with the same background and status line, create separate templates for the background and the status line.
40. Make templates self-contained. Templates should not depend on an individual EB GUIDE model or other elements except datapool items or events. This way templates can be exchanged between EB GUIDE models.
41. An image widget always requires an image resource. If an image widget in a template does not require a fixed image, use a dummy image.

9. Touch input

For more information, see EB GUIDE Studio user guide section "Touch input"

42. If you want a widget to react when the user taps on it, model the action using widget feature **Touch released** and property `touchShortReleased`.
43. To consume a touch event, the corresponding EB GUIDE Script needs to have the return value `true`. It is possible that return values have additional functionality inside EB GUIDE GTF that may influence the behavior.

10. Widgets

For more information, see EB GUIDE Studio user guide section "Widgets".

44. For data that needs to be stored persistently, link widget properties to datapool items.
45. Use the instantiator widget to add or remove widgets at run-time.
    - Use a low number of child widgets as a default setting for `numItems`. These child widgets are instantiated during view creation.
46. When values for `x, y, width, height, visible` need calculation, for example if the x-position depends on the width of a neighbouring widget, update them with separate update scripts. Put the dependent properties into the trigger lists and update the widget property depending on that.