

Using FGAG to export ETS data for Visualization

Cezary Szczegielniak , Markus A. Wischy
cezary.szczegielniak@tuhh.de, markus.wischy@siemens.com

Siemens AG, CT SE 2, Corporate Technology,
Software & Engineering – Architecture,
81730 Munich, Germany

Abstract. *The paper presents a method to analyze the ETS project data according to the FGAG specification. The data is then exported for use in external applications for visualization and control of EIB devices.*

1. INTRODUCTION

An EIB installation can benefit the user much more with graphical control devices (as a PDA, TouchPanel, PC, etc.). In order to make these devices capable of visualization, the configuration data of EIB devices have to be exported for middleware applications.

The ETS project holds the EIB device configuration data of an installation. The paper describes how the export of visualization data is considerably simplified, when the ETS project is created according to the FGAG specification.

The Functional Group Addressing Guidelines (FGAG) is based on a model of pre-defined group addresses for specific functions. FGAG covers functions for lighting, shading, HVAC, security and energy management. The model describes functional entities for various rooms and contained devices (e.g. Room3/Kitchen - Light #1 on/off), as well as global functions (e.g. Whole House – Solar Protection South Level 1).

In this paper an algorithm to analyze ETS project data will be presented. FGAG devices used in the project are recognized, according to their pre-defined group addresses, and validated (e.g. project completeness and sanity checks). These FGAG device data, together with relevant EIB devices data, are then exported for use in external applications.

2. EXPORTING EIB DATA

The EIB is a field-level network for electrical installation control and can be accessed from outside via special EIB devices, which make the protocol conversion at the network border. Those devices connect on one side the EIB and on the other side to the other network (e.g. Ethernet). In this way the EIB devices can be controlled via visualization tools, which can run on a PC or PDA connected to the Ethernet.

In order to be able to control EIB devices from a different network, the controller needs to know the available devices, their addresses and the interrelations. (e.g. the EIB configuration). For EIB installations this configuration is stored (and edited) via the EIB Tool Software (ETS).

There are several possibilities how to extract the ETS data:

- Manually
This is time consuming and error prone, only acceptable for very small installations
- Via an external tool accessing the ETS database
This has two drawbacks: installers need to understand a new tool and the tool itself has hard dependencies on the version of the ETS DB schema, generally not desired
- Via an ETS extension (ETS plug-in)
An ETS plug-in (for an visualization device) has access to the ETS configuration via an API. The installer can use this plug-in (e.g. new device) with ETS to generate the export configuration data.

We choose the third option for generating the export configuration data for two reasons: First, the installer can use the standard tool and second the API provides a more stable basis for software development (reduces maintenance costs for updated versions of the tool).

From the point of view of functionality there are many types of EIB devices, e.g. actuators (dimmers, binary outputs), sensors (switches), devices without communication objects (power supply, choke, etc.). For the visualization tool the important EIB devices are only actuators. Having access to an actuator the application is able to influence a state of the device and thus control e.g. a light. The actuators can also send events, upon their state change, so the status of the device in visualization tool can be updated. One of the key points in ETS project data analysis is to extract only relevant EIB devices. Basically the algorithm should extract only actuators and only those, which have at least one communication object connected to a group address. However there are some more cases when the automatic generation fails.

This main problem is generating an export structure that contains a semantic level usable for a visualization tool. Especially the semantics of EIB devices can not be determined automatically in a generic way, since for each EIB device, the device type, the channel definition and the actual location need to be known (and this is not stored in the ETS manufacturer DB). Besides the missing data, also a standardized definition of these semantics is needed (e.g. manufacture independent definition).

Here the FGAG definition (see below) solves these problems and allows to define a mostly automated process for the extraction of data needed for visualizations.

3. FGAG

3.1. General description

The Functional Group Addressing Guidelines (FGAG) [see 1] offers a pre-defined set of group addresses, which covers functions for lighting, shading, HVAC, security, and energy management.

The model for allocation of functions to pre-assigned group addresses assumes that a residential environment contains spaces and devices, which are called *functional entities*.

- Example from the FGAG specification:
”A residential dwelling may have a front yard, a back yard, the house itself with basement, ground floor, second floor, and attic. There are a number of rooms e.g. living room or kitchen. Each of these rooms is a functional entity.”

Spaces and devices contain *functions* specific to each individual functional entity.

- Examples:
Room 3 (Kitchen) Light #1 on/off. A function of the functional entity “Room3”. Date and Time are functions that apply to the whole house. Hence they are functions of the “Whole House” entity.

3.2. Structure of the FGAG

Usually a group address in EIB specification consists of three parts: main address, middle address and sub group address. In the FGAG specification main and middle addresses are used to distinguish different functional entities (e.g. Whole house, Entrance hall, Living room ...).

Within those functional entities the sub group addresses are used to distinguish different functions (e.g. Light #1 on/off, Light #1 dim up/down, Shutter #1 up/down – see following figure: main and middle group addresses in the FGAG specification).

Most of the functional entities have the same feature set, e.g. the same sub group address layout describing the FGAG functions and devices. Those functional entities are between address range 1/3 (Entrance Hall) and 4/2 (Technical Room), where the first digit denotes the main group address and the second digit denotes the middle group address. Those addresses are marked on the figure as a *begin address* and an *end address*.

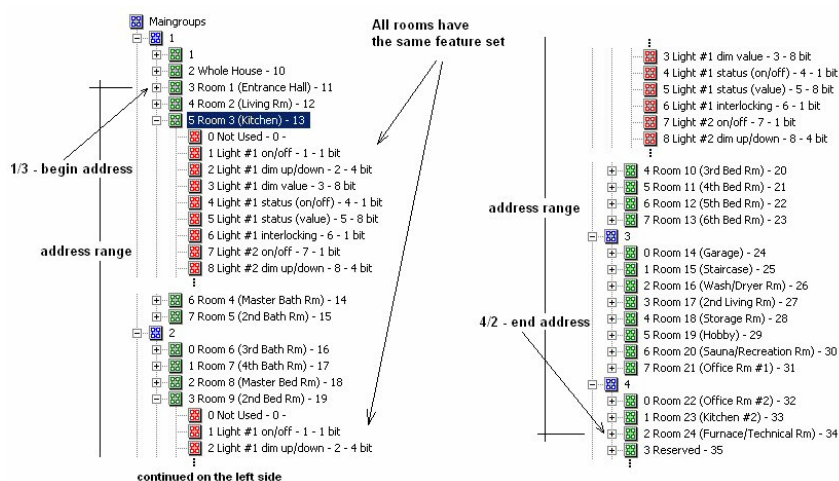


Figure 1: Main and middle group address in the FGAG specification

3.3. Benefits from the FGAG

The standard and well defined structure of the FGAG specification can simplify the process of automatic data generation and device recognition. The recurrent FGAG structure is easy to store and than used to analyze ETS project data. Each group address in the FGAG has well defined function (e.g. 1/5/1: Room 3 (Kitchen) – Light #1 on/off). In the algorithm the project is scanned for FGAG devices and functions, what in practice means that the algorithm has to check which FGAG group addresses are connected to communication objects. In this case there is no problem with device and function recognition (e.g. which communication objects belong to which channel or which device is an actuator). Every function and FGAG device is explicitly defined and localized.

4. ALGORITHM

The following figure depicts the algorithm for generating the exported configuration file of EIB visualization data.

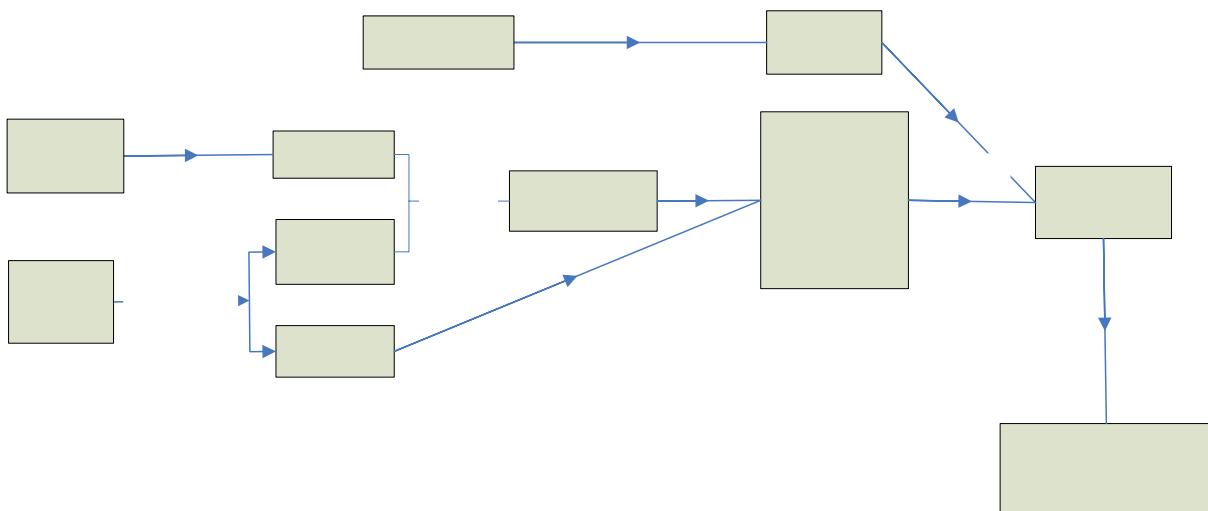


Figure 2: Overview of the export algorithm

Short description of the algorithm steps:

1. FGAG specification is read from the file (FGAG_devices.xml) and saved as data structure in the application. The file contains the generic description of the FGAG functions and devices. The structure will be later used to check which FGAG devices are used in an ETS project. Storing the specification in an external file makes easier to support new FGAG versions (e.g. new functions defined).
2. The ETS project is analyzed. All the EIB devices and group addresses used in the project are extracted and stored in data structures for further processing. Only EIB devices with at least one relevant communication object are considered. A relevant communication object is an object that:
 - is connected to a group address
 - has a communication and transmit flag set,
 - has the write or read flag set (write flag to change the status, read flag to read the status of a communication object).

3. The FGAG – export mapping data is read from the file and saved as data structures. Those data will be later used as a mapping pattern to generate export device list from FGAG device list.
4. Based on the FGAG specification and Group address list find all the FGAG devices used in the project. An FGAG device is used in the project when at least one of its FGAG functions is used (e.g. a group address associated with this FGAG Function exists and has a communication object connected). Data are saved as an FGAG Device list.
5. Check if FGAG devices used in the project are correctly defined (correct group addresses, associated communication objects ...). Save all errors and warnings.
6. Based on checked FGAG device list and FGAG – export mappings create corresponding export devices.
7. Based on export device list generate the EIB – export XML Configuration File.

4.1. Recognize FGAG devices

The visualization data will be later extracted based on the FGAG devices found in this step. The proposal algorithm to recognize FGAG devices used in the ETS project is as follows:

Iterate over **Group Address List**

Look up function_type and device type (from FGAG Device Specs)
 (e.g. check which device type applies for the group address, check which functions matches)

if the address exist in **FGAG Specification list**

if device is not in FGAG device list

create FGAG device

add FGAG function to the FGAG function list associated with this device

else

add FGAG function to the FGAG function list associated with this device

It is important to note, that there might be more than one FGAG device which matches the given address, e.g. Dimmable Light and Binary Light (functions of Binary Light is a subset of Dimmable Light). In this case check all the group addresses of the FGAG functions belonging exclusively to Dimmable Light. If at least one address exists in **Group Address list**, a Dimmable Light is created; otherwise a Binary Light is created. See example below for explanation.

4.2. Check devices

Check previously created FGAG device list for consistency. The FGAG devices must be correctly defined (correct group addresses, associated communication objects, ...). During this step the status of FGAG devices is set to either correct, error or warning. Only visualization data from devices with correct or warning status are extracted.

Check whether an FGAG device has an ERROR status:

- For each FGAG function from the specification of an FGAG device there must be corresponding group address associated with at least one single device
- for each FGAG device there must be at least one EIB device which communication objects are connected to all function of an FGAG device. Moreover each object should be connected only to one function (see the figure below).

Function	group address			EIB Dev. 1	EIB Dev. 2	EIB Dev. 3	EIB Dev. 4
Not Used	1	3	0				
Light #1 on/off	1	3	1	○	●	○	○
Light #1 dim up/down	1	3	2		●	○	
Light #1 dim value	1	3	3		●	○	
Light #1 status (on/off)	1	3	4	○	●	○	
Light #1 status (value)	1	3	5	○	●	○	
Light #1 interlocking	1	3	6				
Light #2 on/off	1	3	7				

Communication object connected to FGAG functions (circles on the right symbolize communication objects; the marked circles symbolize different object, which at the same time belong to the same EIB device).

- For each FGAG function there must be one communication object which:
 - belongs to the set of communication objects described above (each object in the set belongs to the same EIB device),
 - the flags of a communication object correspond to the type of an FGAG function (e.g. for the FGAG function type “set” – communication object must have flags: “communication”, “write”).
- If above rules are not fulfilled – the FGAG Device has an **error** status.
- If the type of the object doesn't correspond to the type of the FGAG function – **error** status, e.g. type of communication object 1 bit, type of FGAG function – 1 byte.
- Each error generates an error description.

Check whether an FGAG device has a WARNING status:

- One Communication Object should be only in one group address. If this rule is not fulfilled – the FGAG Device has a **warning** status.

Function	group address			
Not Used	1	3	0	
Light #1 on/off	1	3	1	○●○
Light #1 dim up/down	1	3	2	○○○○
Light #1 dim value	1	3	3	○
Light #1 status (on/off)	1	3	4	○○
Light #1 status (value)	1	3	5	○○○
Light #1 interlocking	1	3	6	
Light #2 on/off	1	3	7	●○
Light #2 dim up/down	1	3	8	○
Light #2 dim value	1	3	9	○○○
Light #2 status (on/off)	1	3	10	
Light #2 status (value)	1	3	11	

The same object (marked on black) is connected to two group addresses.

- If the flag of a communication object does not correspond to the type of an FGAG function – **warning**.
- If more than one communication status object is connected to the FGAG function (function which is of type “get” - the object with the “read” flag set) - **warning**
- If there are more FGAG function associated with an FGAG device than it results from FGAG specification (e.g. a binary light has one additional function which belongs to dimmable light) – **warning**.
- Each warning generates – warning description.
- Exception: general function (e.g. all lights on/off)

An FGAG device has a CORRECT status:

- The device is **correct** if during the check it got neither ERROR status nor WARNING status.

5. REALIZATION

A plug-in for the ETS3 software is made an example of the algorithm realization. The plug-in is implemented in the ETS as an EIB device (the visualization device). The plug-in generates an XML configuration file, which contains the visualization data.

The configuration file, which is basically a mapping file between EIB and UPnP devices, is used by a middleware called EIB-UPnP Bridge. The bridge is a software implementation which makes EIB devices reachable in a network by the UPnP protocol. In this way all graphical control devices, which implement UPnP protocol (in particular a functionality of a control point), are able to control EIB devices.

6. DIFFERENT MAPPING SUPPORT

To perform mappings, the internal algorithm, based on mapping XML file, generates the export data structure. This data structure is ready to use for a specific visualization tool. Since there can be many visualization tools, the mapping formats supported by them might differ. The presented plug-in generates an EIB – UPnP XML mapping file for the middleware software – EIB-UPnP-

Bridge. In this specific implementation the Export data format is fixed to UPnP and is useful only for this application.

However with little changes other Export data formats can be supported. First a new mapping XML file has to be provided. The file specifies mutual relations between FGAG devices and export data. Second a new mapping algorithm and file generator have to be implemented.

7. OUTLOOK

As a next step the algorithm can be extended for better support of different configuration files. In this case the algorithm will produce a standardized export file. At the end an XSLT will be used to produce specific configuration files. Support for a new data format can be easily added by providing a new XSLT style sheet.

8. REFERENCES

1. FGAG specification, a proposed EIBA standard