Data flows to and from the base station

Deliverable 6.3.2

Planned date M12

Actual delivery:

Draft: 2007-Nov-28

Final: 2007-Dec-21

Version	1.0
Status	Final version
Date	2007-Dec-21

Contributors:

SA: Jeremi Gancet, Michel Ilzkovitz
SHU: Amir M. Naghsh, Chris Roast
HNI-UPB: Andry Tanoto

General Information

Deliverable	6.3.2: Data flows to and from the base station
Workpackage	6.3: Base Station - Human Interfaces
Associated tasks	T6.3.1
Due date	M12
Involved partners	SAS, SHU, HNI

Document Workflow:

Draft version	Jeremi Gancet	Date	2007-11-28
First revision	<Deliverable Leader>	Date	2007-XX-XX
Final Revision	<Workpackage Leader>	Date	2007-12-21
Submitted	<Project Coordinator>	Date	2007-XX-XX

Document History:

Version	Date	Author(s)	Description
0.1	2007-11-28	Jeremi Gancet	Initial draft
1.0	2007-12-21	Jeremi Gancet	Final release

Table of Contents

Table of figures

Table of acronyms

BSC	Base Station Core
GIS	Geographical Information System
HCM	Human Crew Member
HMI	Human Machine Interface
IC	Incident Commander
I/F	Interface
MDRD	Mission Data Recording and Dispatching
MPSEM	Mission Planning, Scheduling and Execution Monitoring
MTE	Mission Template Editor
OC	Operations Commander
SAD	System Architecture Document
SC	Sector Commander
SDP	Sensor Data Processing
SO	Safety Officer
SLAM	Simultaneous Localization and Mapping
TC	Telecommand
TM	Telemetry
URD	User Requirements Document

1 Introduction

The purpose of this document is to define the main streams of data between the base station and the other components of the Guardians system. This requires to identify interfaces, communication flows, and as far as possible the content of the flows.

We firstly remind the design of the base station architecture, as introduced in the SAD. Then we focus on relationship between the components, and accordingly the data flows to consider. We provide preliminary definition of the control flow protocols as far as communications between the base station and the robot's swarm is concerned. We finally introduce latest issues related to the communication between the base station and human crew members on the field.

2 Base station architecture: overview of the components and data flows

2.1 Base station architecture: big picture

The diagram hereafter depicts the overall base station architecture in Guardians, with a focus on the functional components and the data flows between those components.

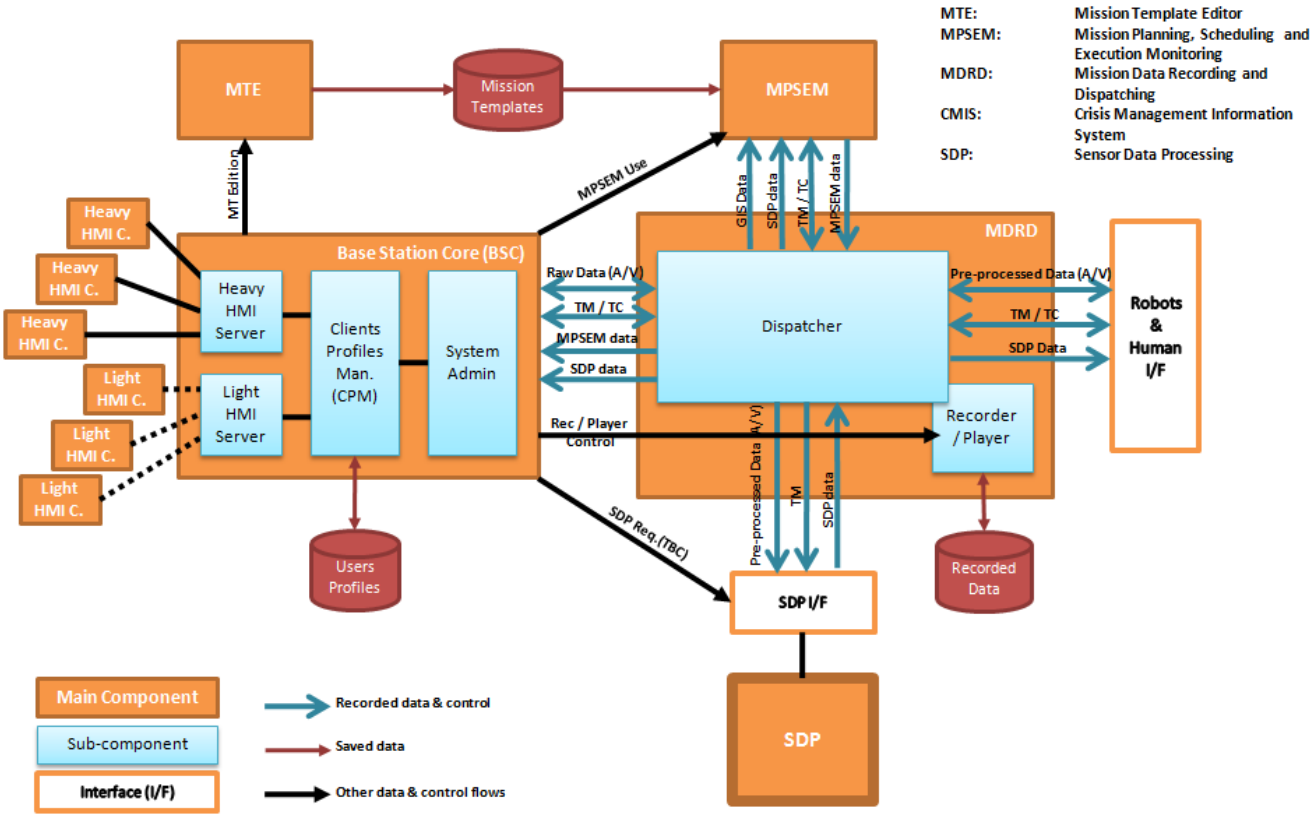

Figure 1: Guardians base station functional architecture overview

We provide in the following a summary of the components description (for more detailed information, please refer to the Guardians SAD). Then the section 3 will focus on the data flows.

The following components have been identified as necessary in the base station:

1. HMI Clients
2. Base Station Core (BSC)
3. Mission Template Editor (MTE)
4. Mission Planner, Scheduler and Execution Monitoring (MPSEM)
5. Mission Data Recorder & Dispatcher (MDRD)
6. Sensor Data Processing (SDP)

2.2 (1) The HMI Clients

HMI clients are the interfaces to the system users: they connect to the HMI server (described hereafter) in order to offer access to the base station, whatever the needs of the users. When an user connects, providing login and password, his profile is restored with rights / accreditation depending on his role.

All HMI clients feature visualization services, but additionally, depending on the clearance (role) of user, certain HMI clients can be endowed with various interaction means such as: screen touching, simple mouse using, joysticks, microphone and speakers, voice recognition, etc. Relevance of these technologies will be carefully studied according to the roles requirements, ergonomics issues, and efficiency in monitoring and controlling the system.

Data visualization / display, personalization (intelligent User Interface) of the content is to be featured. This deals with selection of information to display, respective localization and size in the display, and maybe automatic information changing (or notification) according to events occurring. This shall never be an unwilled constraint for the user (nobody wants jumping trombones anymore...). The possibility to automatically personalize the display according to the use of users is also a topic that will be investigated: for instance learning sequences of actions performed by the user, and offer shortcuts to perform them in a more efficient way.

HMI Clients may be either "lightweight" interfaces, providing mainly passive monitoring features, or "heavy" interfaces, featuring various interaction modalities in addition to monitoring. Lightweight interfaces are likely to be remotely exploitable with a web browser through an internet connection, while heavy ones are likely to require the installation of specific software, and would be probably wire-connected in the vicinity of the base station hardware, for communication efficiency (and security) issues.

2.3 (2) Base Station Core (BSC)

This component is quite central in the system: it provides management means for the overall system, including users administration, system elements health status and testing, software maintenance (updating software components...), etc. It also has two different HMI servers: one related to Heavy HMI service (as a gateway for data from and to the connected Heavy HMI), and the other one related to Lightweight HMI service (as a web server, for access through the internet). The BSC accepts connections with clients HMI, either remotely (lightweight client through internet) or on-site (heavy, full-featured client). These servers allow using the different base station components features (MTE, MPSEM, MDRD...), in addition to enabling monitoring and control interfaces for robots and human crew members operations.

Accordingly the BSC gathers the following components:

- Heavy HMI server: in charge of the heavy HMI clients connections and services providing.
- Lightweight HMI server: in charge of the light HMI clients connections and services providing.
- Clients' profiles manager: manages user profiles, clearance levels, individual user preferences, etc.
- System administration tools: set of tools to administrate the overall base station software.

2.4 (3) Mission Template Editor (MTE)

The MTE provides the tools to edit mission templates that can be used afterward in operational contexts. The possibility to design before-hand such mission templates allows thinking in advance, without the pressure of the operations, about recurrent, classical situations and scenarios of operations. Typically we can imagine a mission template related to the intervention in a warehouse: the typical mission will be mostly indoor, with moderately cluttered (depending on the extent of the disaster...) environment to navigate, with pretty poor visibility and potentially significant risks for human beings. The tasks consist of sweeping the warehouse in order to search for signs of life, to localize host spots and to identify possible danger for firemen. In such a template, different zones have to be delimited. Maps of the area can help in the work, but it is likely that the disaster will make it impossible to simply rely on the maps.

A mission template typically consists of:

- The selection / definition of a particular type of environment (outdoor / indoor / mix, structured / unstructured, temperature characteristics, expected soil characteristics, expected atmosphere characteristics, known presence of humans or not, expected visibility / constraints on sensors, etc.)
- Preferred team (robot platforms + humans) composition for the mission
- Mission steps / main guidelines (kind of sub-missions within the mission, with associated partial order, according to well known procedures for this kind of mission context). Multiple possible schemes will be identified here.
- Selection of possible tasks that may have to be performed in such a mission, and pre-calibration of these tasks according to available (general) knowledge related to this kind of mission context: expected size of area to cover if approximation is possible, stringent time constraints or not, special care needed for detection or not, communication relay needed or not, map likely to be available or not (if not, SLAM needed?...), etc.

Once edited, mission templates are saved in a mission template database which management is under responsibility of the base station coordinator. Mission template can also be simply uploaded from an external source (such as the DMAP), and available mission templates can be shared (saved to an external data support) as well.

2.5 (4) The Mission Planning, Scheduling and Execution Monitoring (MPSEM)

The MPSEM gathers several features related to mission plan execution in an operational context. It first provides the means to instantiate a mission template in the current operational context. For that purpose it supports the base station coordinator in performing the instantiation, such as:

- identifying actually available resources,
- matching the selected template information with available GIS data of the region (extent of the main operational area, buildings, regions where human life is supposed to be, potentially dangerous / hot spots, forbidden zones, unknown zones, etc.),
- selecting high level tasks in the frame of the pre-identified mission (and "sub-missions") steps, according to latest available knowledge about the operations to be performed,
- setting a timeframe for the different steps of the operations (time constraints).

Then, with such an instantiated mission, the MPSEM may generate a detailed task plan, and accordingly may schedule the activities according to available robots, human crew and sensors, and taking into account identified time constraints. The result is a consistent task plan ready for execution by robots, human crew team and base station team.

It is important to mention that the MPSEM will usually simultaneously manage multiple robots and humans, hence coordination of activities is an important point to consider, probably to be tackled at the scheduling level: indeed this is about ensuring that resources are properly allocated, without conflict. However contingencies necessarily occur during execution: one shall not rely on pre-established plan. Instead one should be prepared to revise (or "repair") task schedule or task plan, according to the impact of the contingencies.

2.6 (5) The Mission Data Recorder and Dispatcher (MDRD)

The Mission Data Recorder and Dispatcher (MDRD) is twofold: its main role is to listen to and save all information transiting from the base station, to the base station, and in a certain extent, within the base station: controls flows (requests towards both human crew members and robots, execution status, system related controls flows...) and data flows (sensors data TM, housekeeping TM, raw video and sound data, etc.). In addition, considering the need to have all transiting data dumped through the MRDR, it makes sense to use it as a node that gathers data from and dispatches data to all components of the base station (and also from and to components outside the base station, through adequate interfaces).

Alternatively the MDRD can "replay" (some of the) recorded data as if it was current data transiting in the system: this could be used to debrief recorded mission through the same monitoring devices (HMI) as used during actual operations (but obviously without the ability to interact).

2.7 (6) The Sensor Data Processing (SDP)

This component gathers the functions to process the incoming sensor data from the robots in order both to support the localization of the robots (SLAM supported by exteroceptive sensors) and to perform multi-modal map building of the environments (2D / 3D obstacle maps, chemical map, etc.) that would be too much CPU-consuming for onboard performance. Resulting data are both sent back to the robot (when the communication bandwidth makes it possible), and provided to the system users (in support to robots operations) and to the stakeholders for further decision making with the disaster management authorities. This component is to be designed and implemented in the frame of WP6.2, thus we do not provide more detail about his features in the current deliverable.

3 Preliminary identification of data flows

3.1 Data flows with the robot swarm

3.1.1 Base-station => robot swarm

According to the “big picture” of the base station in section 2, the following data flows have been identified from the base station toward the robot swarm:

- TC flow: this deals with all the commands that may be sent from the base station to the robots of the swarm. It may typically be consist of a structure of data containing an unique, incremental ID attached to the request as a reference, a command ID referring to the command to be sent to the robot(s), and a number of parameters, depending on the command (see section 4 for more information about foreseen protocols).
- SDP data: the purpose is to transmit back to the robot relevant information about environment models, localization of robots and / or human, as processed in the base station within the SDP. The robots may use such information to better self-organize, update their plans, and thus perform more efficiently.
- Audio / Visual flows: this is a possibility that is still being evaluated. The purpose would be to “project” on the field audio or video flows of data, typically emitted by the users of the base station, through the robots.

3.1.2 Robot swarm => base station

- TM flow: this consists of both task execution and health status information, and acquired data through the robots’ sensors: chemical, visual, ultrasonic, etc.
- Audio & video: robots having cameras and microphones may acquire and transmit rough video and audio data to the base station. These information may either be provided -as is- by the base station users, or possibly be refined and merged with other information in the SDP.

3.2 Data flows with the human crew members

3.2.1 Base-station => human crew members

As for robots, there is a need in Guardians to transmit information to the Human Crew Members (HCM) on the field. Here are the identified data flows:

- TC flow: as for the robots, this deals with all the commands that may be sent from the base station to the human crew members. It may typically gather information such as a request reference number, a reference to the command to being requested, and a number of parameters, depending on the command (see section 4 for more information about foreseen protocols). Command sending will very likely not be formulated in the same way as the ones addressed to the robot, but rather be expressed in a natural language form.
- SDP data: the purpose is to send to the HCM relevant information to support them in localizing themselves, identifying objects and places in the environment, and taking decisions
- Audio / Visual flows: the purpose would be to provide to the HCM audio and video data purposely emitted by the base station user. Regarding audio data, this is a “phone”-like feature which could obviously make sense, although other issues shall

be considered, such as cognitive load it would generate. Sending visual / video data to the HCM may also in some extent be considered, but once again the impact on the HCM attention shall be considered. Moreover, bandwidth in Guardians is a critical issue that may drastically limit the use of such a data flow.

3.2.2 Human crew members => base station

Human may have to send information to the base station too, either messages or data acquired by devices they carry.

- TC flow: HCM may want to send request to the base station, in the same way as the base station shall be able to send commands to the HCM. It makes sense to consider such a possibility for human on the field in Guardians, for instance to ask for support (human or robot). One can argue that robot could as well take the initiative to emit requests toward the base station: this is true, but the purpose of Guardians is rather to have a number of limited capabilities robots, hence not able to individually take decision out of their own scope. Collective decision making could be performed, but the base station allows alleviating this issue.
- TM flow: HCM will carry a certain number of sensing devices (possibly monitoring their health or sensing the world around), which acquired data will be transmitted to the base station. This will be part of the TM flow.
- Audio and video: besides the TM related data, rough audio or video may be acquired from devices carried by HCM. Rough video or audio data may be transmitted to the base station, shall the communication bandwidth allow it (which can be the case at some time, and not anymore the case at other moments).

3.3 Data flows with the SDP (inner base station)

We provide here some details about the data flows for the SDP, although it is part of the base station. The reason is that the SDP is developed in the frame of the standalone task, by different partners, and thus clarifying the related interfaces is definitely relevant.

3.3.1 SDP interface => SDP

The SDP needs data acquired by robots and HCM sensors on the field, in order to merge information into actually exploitable and useful models: this can be 2D / 3D models of the environments, chemical maps, localization of the robots and humans, etc. For this purpose, the SDP interface needs a number of data in inputs:

- All possible TM data, ranging from picture acquired by the robots, to chemical products concentrations at different locations, through information about communication topology / connectivity between the robots (which can indeed be useful in building a map of the environment).
- Rough video and audio: it is still to be decided whether such data may be relevant in elaborating maps and models within the SDP. Nevertheless, the availability of the data flow is made possible.

3.3.2 SDP => SDP interface

The SDP will return different models such as 2D / 3D maps of the environment, that both the base station users and robots (and maybe also human crew member on the field) will be able to exploit. Further refinements of the SDP components and features will give additional keys on the nature and modalities of use of such a data flow.

4 Control flow protocols: a first draft

We provide in this section a draft of the protocols that are likely to be used regarding control of the robots and human crew members. We do not significantly detail the “status” data flows, as they still require work on unification of what is to be provided from the different entities in Guardians.

4.1 Individual robot housekeeping commands

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
10	sleep (soft stop)	-	Get the robot in the sleeping state	error_code: 0	error_code: -1 (refine error code ?)
20	wake_up	-	Get the robot in the active state	error_code: 0	error_code: -1 (refine error code ?)
30	hard_stop	-	Get the robot hard-stopped	error_code: 0	error_code: -1 (refine error code ?)
40	power_up (after hard_stop)	-	Powers-up the robot, get the robot in the asleep state.	error_code: 0	error_code: -1 (refine error code ?)

4.2 Individual robot motion commands

These commands deal with the actuation of the robots, i.e. essentially motion. Some of these commands are rather adapted to holonomous robots, while other better fit non-holonomous ones. We suggest to let open the possibilities to control a robot, and to always have counterparts in commands expression in such a way that it’s possible to convert one into another.

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
100	thrust	arg1: float: +/- *percentage*	Activate robot's thrust to the specified level	error_code: 0	error_code: -1, stat_1: current thrust
105	move	arg1: float: +/- *distance in meters*	Makes the robot move in the default / current direction. Positive makes a move forward, negative makes a move backward. With non-holonomous robots, set the steer to 0 first.	error_code: 0	error_code: -1, stat_1: distance moved
110	steer	arg1: float: +/- *percentage*	Non-holonomous robot: set the steering to the specified value. No actual turn (thrust needed). Positive makes it steering left, negative makes it steering right.	error_code: 0	error_code: -1, stat_1: current steering
115	turn	arg1: float: +/- *degrees*	Holonomous robot: Make the robot turn by the specified angle. Positive makes it turning CCW, negative makes it turning CW.	error_code: 0	error_code: -1, stat_1: achieved turning
120	goto_single	arg_1: point: *target loc*	Makes the robot move to the targeted point.	error_code: 0,	error_code: -1, stat_1: current loc
130	goto_list	arg_1: int: *nb of points* arg_2: xyz: *point1* arg_3: xyz: *point1* ... arg_k: xyz: *pointk*	Makes the robot moves through each of the specified points, in the right order.	error_code: 0,	error_code: -1, stat_1: latest reached point stat_2: current loc
140	wait	Arg1:float: time in s	Makes the robot wait before to start another task (*)	error_code: 0	error_code: -1 (refine error code ?)
150	resume (when waiting)	-	(*)	error_code: 0	error_code: -1 (refine error code ?)

(*) The “wait” and “resume” commands (140 and 150) shall be considered in the case where a robot is able to receive and manage a (partially ordered) set of tasks to be executed (i.e. a task plan). If a robot can only process a single task at a time, then the “wait” and “resume” commands are useless (indeed when the robot doesn’t have any task to perform, it’s equivalent to waiting).

4.3 Individual robot sensor commands

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
200	switch_on_sensor	arg_1: int: *sensor id*	Switches a targeted sensor on.	error_code: 0	error_code: -1 (refine error code ?)
210	switch_off_sensor	arg_1: int: *sensor id*	Switches a targeted sensor off.	error_code: 0	error_code: -1 (refine error code ?)
220	acquire_data	arg_1: int: *sensor id* arg_2: int: *acq mode* (0) -> single acq / shot (1) -> continuous acq (2) -> timed acq (arg_3: int: *duration (ms))*	Starts data acquisition with the specified sensor.	error_code: 0	error_code: -1 (refine error code ?)
230	stop_acquire_data	arg_1: int: *sensor id*	Stops data acquisition with the specified sensor.	error_code: 0	error_code: -1 (refine error code ?)
240 (**)	point_sensor	arg_1: int: *sensor id* arg_2: int: pointing type (0) -> point coord (1) -> vect coord arg_3: xyz: coord	Orients the sensor toward the specified direction. (**)	error_code: 0	error_code: -1 (refine error code ?)

(**) Command 240 is useful only for sensors with orientation capabilities (if any), such as a pan-tilt mounted camera bench.

4.4 Group of robots, “high level” commands

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
500	make_com_mesh	arg1: map: *area to cover* arg2: int: *quality level*	Robots take position for a valid communication network mesh.	error_code: 0	error_code: -1, stat_1: current thrust
510	explore_and_map	arg1: map: *area to cover*	Makes the robots share the exploration & mapping of the zone.	error_code: 0	error_code: -1, stat_1: current steering
520	patrol	arg1: map: *area to cover* arg2: int: *duration*	Makes the robots patrol (i.e. continuously sweeping) a given zone.	error_code: 0,	error_code: -1, stat_1: current loc
530	follow_fireman	arg1: int: *target fireman id*	Robots follow target fireman.	error_code: 0,	error_code: -1, stat_1: latest reached point stat_2: current loc
540	guide_fireman	arg1: int: *target fireman id* arg2: point: *target loc*	Robots bring target fireman toward target loc.	error_code: 0	error_code: -1 (refine error code ?)
550	select_formation	Arg1:enum:*formation type:* 0 = Line 1 = Square 2 = Diamond 3 = Circle ...	Robots adopt the selected formation type	error_code: 0	error_code: -1 (refine error code ?)
560	abort_mission	-	Robots fast escape (what about humans still on the field ?)	error_code: 0	error_code: -1 (refine error code ?)
570	send_backup_robots	arg_1: int: *nb of points* arg_2: xyz: *point1* arg_3: xyz: *point1* ... arg_k: xyz: *pointk*	Robots organize to send available (backup) ones to the specified destinations.	error_code: 0	error_code: -1 (refine error code ?)

4.5 Human Crew Members commands

4.5.1 Base-station requests to the human crew members

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
300	abort_mission	-	The mission is interrupted, but no immediate risk is identified.	- OK	- REPEAT - NOT POSSIBLE
310	exit_asap	-	HCM shall leave and find an exit as soon as possible	- OK	- REPEAT - NOT POSSIBLE
320	explore_current_space	-	HCM shall explore the location where he is.	- OK	- REPEAT - NOT POSSIBLE
330	message	*arg_1: string: msg_ content*	HCM shall receive and understand the message	- OK	- REPEAT - NOT POSSIBLE
340	follow_robot	*arg_1: robot_ID (color ? what else ?)*	HCM shall follow the specified robot	- OK	- REPEAT - NOT POSSIBLE

4.5.2 Human crew members requests to the base station

ID	Name	Args	Description / Effect	Returned Status if Success	Returned Status if Failure
400	master_alert	-	The HCM notifies a critical issue, that activates a master alarm in the base station	- OK (+ msg with detail)	-
410	need_cares	-	The HCM notifies that he needs cares	- OK (+ msg with detail)	- NOT POSSIBLE (+ msg with detail)
420	send_me_squad_members	*arg1: int: nb of SM*	The HCM asks for human support	- OK (+ msg with detail)	- NOT POSSIBLE (+ msg with detail)
430	send_me_robots	*arg1: int: nb of robots*	The HCM asks for a certain number of robots	- OK (+ msg with detail)	- NOT POSSIBLE (+ msg with detail)
440	prepare_emergency_vehicles	*arg1: int: nb of victims*	The HCM requests emergency vehicles to get ready and in stand-by (victims may have been located)	- OK (+ msg with detail)	- NOT POSSIBLE (+ msg with detail)

5 Discussion on base-station ⇔ human crew member data flows

We provide in this section consideration related to discussion with Guardians end-users, i.e. SyFire firemen. Some points have been introduced earlier in the URD deliverable (D1.1/2), and some other are further related thinking.

As a baseline for discussion, the following incident structure has been studied as a reference incident management structure:

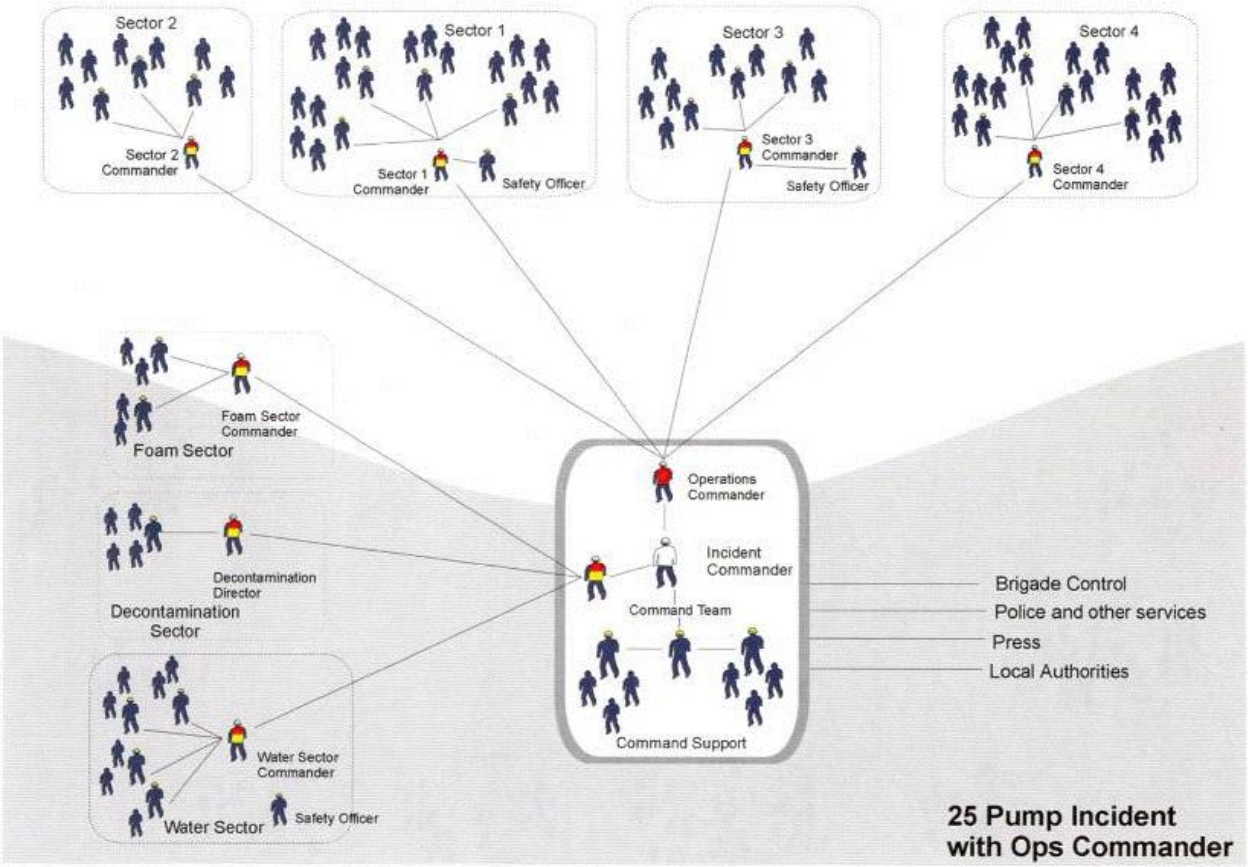

Figure 2: Incident management structure (courtesy of SyFire)

A number of firemen roles are identified through this scheme. Let mention the following:

- Incident Commander (IC)
- Operations Commander (OC)
- Sector Commander (SC)
- Safety Officer (SO)
- Firemen on the field, or “Human Crew Member” aka. HCM (notion defined in the frame of Guardians)

Taking into account the variety of roles to consider in the firemen hierarchy during operations, we sketched up the diagram hereafter, where the communication links between the base station and the firemen (according to their roles) is highlighted:

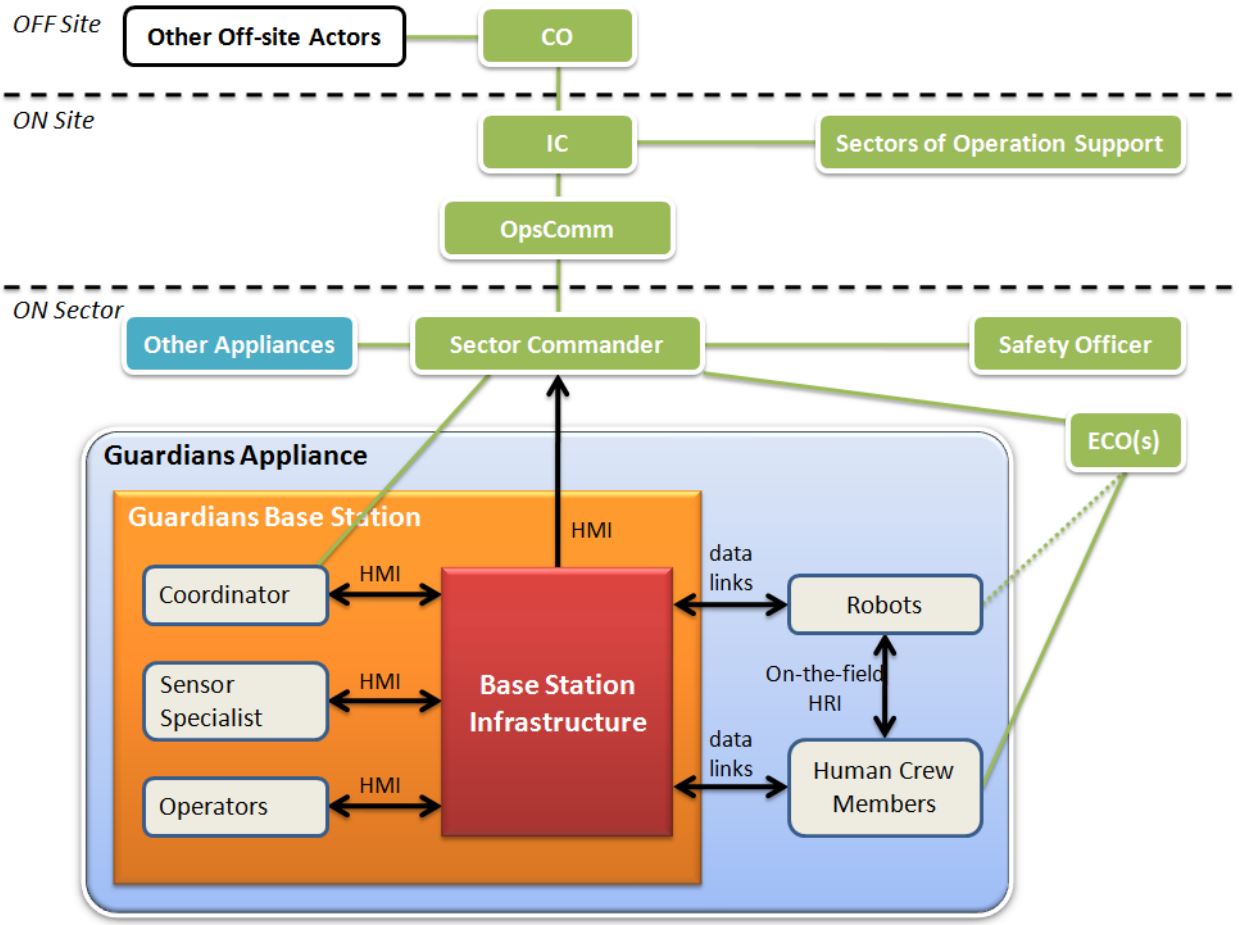

Figure 3: Firemen roles w.r.t. the base station infrastructure and users

An essential concern that arises in Guardians is the possibility to interact between the base station and the human crew member or not, and in which extent. Considering the functionalities that the base station is going to provide in Guardians, it seems appropriate for the fire fighters in the field to be able to have direct contact with an operator within the base station who can act as the command support officer to the incident commander.

However, it can have some complications for the fire fighters in the field by requiring them to interact with two different commanders; one in the base station and one at the firemen control center. In addition to communications with OC as part of the chain of commands, the fire fighter would need to maintain his contact with the base station to assign robots with more sophisticated tasks. Handling too much of communication could be problematic for the HCM in the field of incident.

Therefore, it is beneficial to identify what interaction channels (Audio/Visual/Tactile) would be used for each of the requests, and who in the chain of commands would send or receive these requests.

For instance, according to what is suggested in the URD, fire-fighter in the field can use a tactile interface on his/her gear to send the requests 400, 410 and 430 (see section 4.5.2) which can be visually displayed at the base station (or the OC). While the fire-fighter cannot use the simple tactile interface to make requests 420 and 440 and an audio channel is more recommended to make such requests.

The requests 300 and 310 (see section 4.5.1) can be communicated to the fire fighter through a visual channel. For example the LEDs which are used to provide directions can also be used to indicate warning such as emergency evacuation (exit_asap). For instance all the LEDs can go red to indicate that fire fighter has to exit as soon as possible.

Figure 4: Early prototyping of LED activation system, depending on the message to pass to the fireman (by SHU)

For the requests 320 and 330 (see section 4.5.1), the base station probably needs to maintain an audio channel for interacting with fire fighters. For example to send a 320 request, usually a number of critical information is required to be send with it as well, information such as the tactic to be used, the involvement of rooms/floors and so on.

Additionally, firemen are used to exploit a dedicated logging system to mention and communicate events, with a dedicated formalism: this shall definitely be taken into account in the transmission of information from firemen on the field to OC, other relevant roles in the firemen commanding chain, AND the base station as well. This will further be carefully studied with the support of SyFire firemen.

6 Conclusion

We presented in this document the nature, scope and (in a certain extent) the approach to the protocols to be used for the data flows dealing with the base station in Guardians. This work is critical, as it sets the baseline for further interaction between the components developed by the different partners of the project. For this reason, multiple iterations in the refinement of the type and content of data flows have to be considered in order to come to a mature and meaningful data exchange scheme in the Guardians system. As substantial adjustment and progress will be identified in upcoming work, this document will be updated accordingly, in order to serve as an exploitable reference.