Deliverable Reference	:	D5.4
Title	:	Unitary & integrated test plans, and release schedule for the Orbital RI
Confidentiality Level	:	PU
Lead Partner	:	SPACEAPPS
Abstract	:	This deliverable describes detailed unitary and integrated test plans for the Orbital RI, following the detailed specification done in T5.1. Additionally, this task will produce a detailed release goals statement with associated schedule for Orbital RI specific contributions
EC Grant N°	:	730014
Project Officer EC	:	Christos Ampatzis (REA)

InFuse is co-funded by the Horizon 2020
Framework Programme of the European Union

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and cross-reviewed by:	Jeremi Gancet Shashank Gonvindaraj	SPACEAPPS	30/03/2018
	Nassir Oumer Michal Smisek	DLR	
	Nassir Oumer Michal Smisek	DLR	23/01/2018
	Shashank Govindaraj Jeremi Gancet Xavier Martinez	Space Applications	
	Fabrice Souvannavong Vincent Bissonnette Arnaud Moura	Magellium	
	Mark Post Alessandro Bianco Romain Michalec	Strathclyde University	
	Simon Lacroix Quentin Labourey Andrea De Maio	CNRS/LAAS	
	Raul Dominguez	DFKI	

DOCUMENT CHANGE RECORD

Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.1.0	07/12/2017	Shashank Govindaraj	All	Document structure
0.2.0	19/12/2017	Shashank Govindaraj	All	Assignment of sections to partners
0.3.0	02/01/2018	Shashank Govindaraj Xavier Martinez Nassir Oumer Michal Smicek	All	Inserting preliminary inputs to the sub-sections
0.4.0	19/01/2018	Mark Post Vincent Bissonnette Simon Lacroix Raul Dominguez	Chapter 3 & 4	Release schedule, internal unitary and integrated test plans for DFNs and DFPCs
0.5.0	23/01/2018	All partners	Chapter 4 & 5	Testing and validating procedures for DFNs/DFPCs in OG6 facilities
1.0.0	25/01/2018	Shashank Govindaraj Joseph Salini Nassir Oumer Jeremi Gancet	All Chapters	Final check of content and formatting the document
2.0.0	30/03/2018	Shashank Govindaraj Nassir Oumer Jeremi Gancet Xavier Martinez	All Chapters	Addressing RIDs emitted by the PSA

Executive Summary

This deliverable addresses the approach and methods for testing and validating DFNs and DFPCs internally, and within OG6 facilities for the Planetary Track. CDFF core methods are designed and developed to be middleware independent. In order to test the DFNs functionality and fault tolerance, they would expose their interface via a Python interface or be wrapped in a target RCOS to allow testing them internally. The release sequence of the CDFF software components for this reference implementation are highlighted for each of the 3 internal milestones at M18, M20 and M22. The procedure for testing and evaluating each DFN category and DFPCs for Planetary-RI with the OG6 DLR OOS Sim is elaborated.

Table of Contents

List of of Figures

1 Introduction

1.1 Purpose

This document describes the detailed approach for testing and validation of CDFF software Reference Implementations (RI) of the orbital scenario. During the development phase, internal testing of the CDFF components will be conducted in phases (pre-TRR) followed by the final validation tests within OG6 facilities (post-TRR). It includes the preparation and verification of the EGSEs and their interfaces in terms of data and software, the latter being based on the design of the CDFF as described in D5.1.

The objective of orbital RI of InFuse is to demonstrate and evaluate the full capabilities of the CDFF: from space compliance to state-of-the-art algorithms, from traditional to innovative sensors, to demonstrate that the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

Note: This document does not focus on the validation of the supporting EGSEs (i.e. deployment of CDFF on RTEMS-Leon architecture) in the orbital reference scenarios.

1.2 Structure

This document is structured as follows:

Section 1: Introductory material to the deliverable.

Section 2: Overview of unitary and integrated testing approaches

Section 3: Software deliverables release schedule

Section 4: Pre-TRR Internal (to OG3) unitary and integrated testing plans

Section 5: Post-TRR testing within OG6 facilities

1.3 Applicable documents

AD1 InFuse Grant Agreement

AD2 InFuse Consortium Agreement

AD3 InFuse internal management manual for project partners

1.4 Reference documents

RD1 Description of Action document

RD2 D3.1 Technological Review

RD3 D3.2 System requirements

RD4 D3.3 Early CDFF architecture and ICD

RD5 D4.1 Technical Trade-off Analysis

RD6 D4.2 Advanced CDFF architecture and ICD

RD7 D4.3 CDFF Unitary and integrated test plans

RD8 D4.4 Preliminary design document

RD9 D5.1 Definition and specification of the Orbital RI and EGSE

1.5 Acronyms

DF: Data Fusion

CDFF: Common Data Fusion Framework

API: Application Program Interface

OOS: On-Orbit Servicing

RCOS: Robot Control Operating System

DFN: Data Fusion Node

DFPC: Data Fusion Process Chain/Compound

DFNCI: Data Fusion Node Common Interface

DPM: Data Product Manager

MW: Middleware

LOS: Line of Sight

Fps: Frames per second

OOS-sim: On Orbit Servicing simulator

OG: Operational Grant

IMU: Inertial Measurement Unit

OT: Orbital Track

PT: Planetary track

OBC: On Board Computer

DEM: Digital Elevation Model

FPGA: Field Programmable Gate Array

2 Approach for Unitary and integrated testing

This section describes the general approach for testing independent and integrated software component which are a part of CDFF-Core and support that are complemented by the CDFF-development tools. During the life cycle of the project, testing and verification has been distributed into 2 phases - internal to InFuse and final with OG6 facilities. This has been described in D4.3 and in the project DoA documents. The facilities for internal and final, testing and validation for the orbital RI include the OOS-Sim facility at DLR.

2.1 OG3 Internal testing and validation of CDFF

The following section covers the strategies for testing DFNs and DFPCs. A DFPC would be assembled as a single software component consisting of a DFPC controller (Refer to deliverable D5.1) and underlying DFNs. The DFPC controller manages the sequence of triggering data fusion functions and manages the flow of data between DFNs. Hence the DFPC being a monolithic component can be tested with the same approach as DFNs.

Internal testing of CDFF for the RI The testing strategy depends upon the nature of the DFN, i.e. whether (1) the DFN developed from scratch, or is it based on legacy code (i.e. re-using previously existing code), and (2) does the DFN depend on third party libraries.

In case legacy code or third-party libraries are involved, unit tests will be developed in the DFN where these legacy or third party software are exploited (e.g. calls to functions), and shall allow verifying that the interactions with that existing software are robust.

All new code will be requested to come with a comprehensive unit test coverage. Evidences of coverage rate will be requested from contributors, accordingly.

Memory usage profiling : Valgrind profiler will be used to measure the memory usage of DFNs and DFPCs in the developers and target environment (Linux based) along with detecting memory leaks in DFNs and DFPCs. The Valgrind Massif tool is a heap profiler. It measures how much heap memory your program uses and measure the size of the program's stack(s).

Computation profiling: The gperftools CPU profiler has a very little runtime overhead, provides features like selectively profiling certain areas of interest and has no problem with multi-threaded applications. This can be complemented by a free or licensed version of the Intel VTune Amplifier (https://software.intel.com/en-us/intel-vtune-amplifier-xe) for accurate profiling of the computational loads of DFNs and DFPCs.

There are 2 methods for performing functional testing and validation.

2.1.1 Python bindings with data flow control

The DFNs/DFPCs are designed and developed to be middleware independent. In order to test the DFNs functionality and fault tolerance, DFNs would expose its interface via a Python interface (via C/C++ Python bindings) to allow testing DFNs. The DFN Python interface would connect to the data flow control within InFuse which is based on Python Pandas for reading sample sensor data from csv like files and logging the output data.

Figure 1 shows the steps required to expose the C/C++ interfaces of a DFN via an equivalent Python interface. This approach facilitates integration with the data flow control relatively fast to validate the functions and behavior of the DFN logic. The steps are briefly described as follows:

- A C/C++ library (or source code) of the DFN is considered to be developed and available for testing
- The interface for the DFN is available in the header (*.h or *.hpp) file.
- Components of the DFN interface that needs to be exposed is encoded as a Python SWIG interface (within a *.i file)
- The Swig tool chain uses the swig interface file to generate an equivalent Python interface definition file and C++ wrapper file for Python bindings.
- The generated C++ code needs to be compiled and linked to the Python interface definition file.
- In the final steps, a Python test suite program needs to be developed that can include the Python interface definition file and use it to access the DFN interface for testing them with the data flow control.

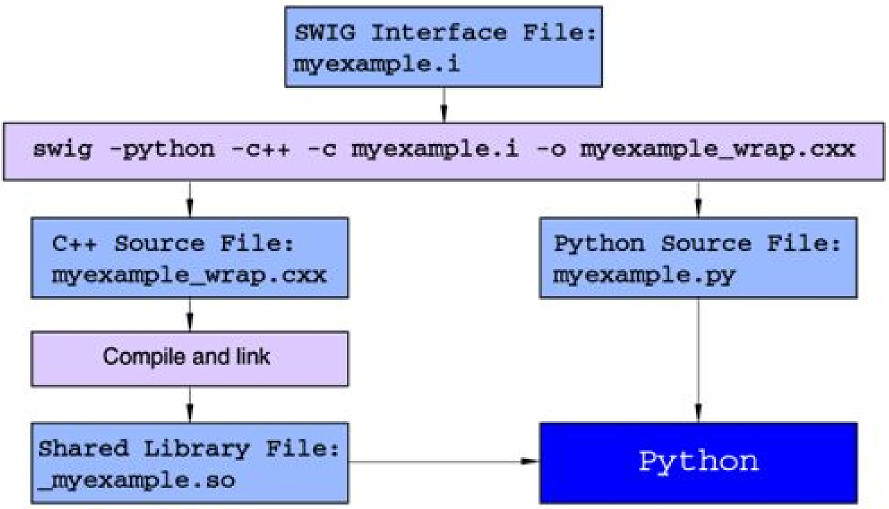

Image 1: Workflow for exposing DFN interfaces through Python for testing off-board

2.1.2 As a robotics middleware component

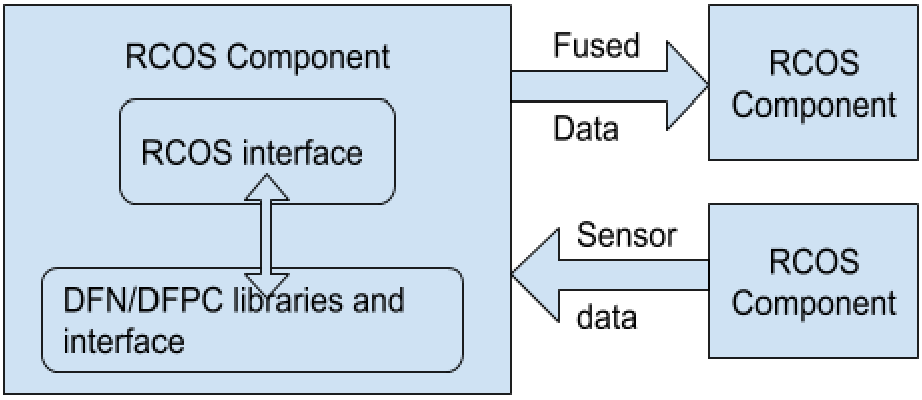

Image 2: DFNs/DFPCs tested as RCOS components

A secondary approach would be to integrate the CDFF-core DFN into a specific robotics control software (RCOS) component and use the log replay mechanisms to access sensor data or data samples that are intermediate to the DFPC. The proposed robotics middleware within InFuse include ROS, RoCK, YARP and GenoM3. This approach would also demonstrate the ability of incorporating DFNs within multiple middleware components

2.2 Testing and validation of CDFF in OG6 facilities

The approach for validating CDFF software components are driven by OG6 testing and validation facilities, associated sensors, software and data interfaces within the representative analog. In the case of InFuse, the DLR OO-Sim facility will be used for final testing and validation activities. DFPCs developed within the OT-RI (reference to D5.1) will be evaluated between M23-24 of the project in DLR OOS-Sim facility. The scenarios and use-cases identified in D5.1 will be elaborated in the following section. The testing procedure would be carried out in an open loop manner i.e commanding manipulators are not influenced by CDFF data fusion outputs. There are 2 approaches for validating CDFF software with open loop operations (i) on-line data acquisition and fusion (ii) on-line recording of data and offline data fusion.

The OOS-Sim facility is equipped with a set of baseline sensors and calibrated in the current configuration (IMU, Stereo-camera). Additional sensors for validating OG3 would need to be mounted and calibrated for InFuse such as 3D Lidar, ToF camera and/or Force/Torque sensor. The OBC for this setup will consist of the existing computer infrastructure at the facility for commanding and acquiring data from the OOS-Sim manipulators that would be interfaced with an additional computer running DFPCs via an ethernet interface.

The DFPCs would be wrapped in the target middleware (typically ROS) and deployed on the OBC to acquire data sensors to perform on-line or off-line data fusion. Parametric and model specifications are required for specific DFPCs that will be addressed in the following sections of this document. Existing sensor data acquisition interfaces will need to be customized for transferring data to the computer hosting CDFF for online data fusion.

2.3 Recap of Orbital RI scenarios

The orbital reference implementation consists of a localization and reconstruction of a target satellite to support rendezvous and an on-orbit servicing. A servicer satellite is assumed to operate within a mid- and close-range so that an attitude and a position of the target satellite is continuously estimated in real time. Hence, the orbital RI addresses the following DFPCs in mid- and/or close-range:

- Target detection
- Target tracking
- Target reconstruction

The detection DFPC globally estimates an initial position and orientation of the target which can be used to initialize a pose tracking. This DFPC is also used for re-initialization in case the local tracking is not succeeded. Unlike the detection DFPC, the tracking DFPC estimates the pose of the target continuously at higher data rate. On the other hand, the target will be reconstructed which can be further processed to produce a geometric model.

2.4 Preparation of datasets

The reference implementation will cover recording of datasets for the validation of the DFPCs. The main data sources for the validation come from stereo/mono cameras and LIDAR. The dataset is recorded under selected representative illumination condition and motion trajectories. Two on-orbit servicing simulators will be used depending on range of operation:

- DLR OOS-sim: for close-range tracking and detection (online or recorded data)
- GMV Rendezvous simulator: for mid-range detection, tracking and reconstruction (on recorded data)
- Strathclyde visual reconstruction and tracking facility in SMeSTech robotics lab (for real models).
- Blender and MORSE simulators: for mid- to close-range generation of photorealistic images to be used for developmental testing of target 3D reconstruction and recognition functions.

On the other hand, stereo image datasets are available which have been used for validation of a visual tracking at close-range. Some of them can be accessible publicly at DLR[1]. The CROOSCV dataset, will be used for validating pose detection DFPCs[2].

[1] http://rmc.dlr.de/download/crosscv-dataset/

[2] http://rmc.dlr.de/rm/en/staff/martin.lingenauber/crooscv-dataset/

3 Release Schedule

3.1 DFPCs (with associated DFNs)

Priority	**USTRATH**
M18 (High)	Feature Detection and Matching
M20 (Medium)	Point Cloud Triangulation and Construction 3D Point Cloud Object Recognition
M22 (Low)	Bundle Adjustment and Optimization

Priority	**MAG**
M18 (High)	N/A
M20 (Medium)	Mid and Close-range Target Detection and Tracking LIDAR-based Target Tracking
M22 (Low)	Far-range Target Tracking

Priority	**DLR**
M18 (High)	NA
M20 (Medium)	Close-range visual tracking on recorded data set
M22 (Low)	Online Close-range visual tracking

Priority	**SPACEAPPS**
M18 (High)	DFPC Haptic scanning
M20 (Medium)	N/A
M22 (Low)	N/A

Priority	LAAS-CNRS
M18 (High)	N/A
M20 (Medium)	N/A
M22 (Low)	N/A

3.2 CDFF-Support and Dev utilities (iterative updates)

This section describes the release schedule for CDFF-Support and CDFF-Dev sub-system components. A list of features or capabilities (based on requirements) that will be developed and released is provided.

Schedule	Orchestrator features or capabilities
M18	• Identify the queries from OG2 and associated parameters • Handle essential queries from OG2 to map into corresponding DFPC • Trigger start and stop of DFPCs based on OG2 query • Monitor and log basic run-time status of DFPCs
M20	• Interface Orchestrator with the DPM at the DFPC level to acquire fused data and forward it to OG2 • Command OG4 sensor operating modes, implement handshaking sequence and monitor error states • Acquire sensor data from OG4 sensors via the mock data acquisition interface
M22	• Interface Orchestrator with the central DPM • Notification of run-time states to OG2 for intervention of unknown states. • Monitor complete DFPC run time and error states and correction (FDIR) • Commanding of all OG4 operation modes and associated FDIR • Running multiple DFPCs simultaneously exchanging data via the central DPM.

Schedule	DPM features or capabilities (local to DFPC and global)
M18	• Position Manager DFPC: ability to fuse Wheel Odometry and Visual

	Odometry
M20	• Update of Position Manager DFPC: ability to fuse SLAM pose estimates
M22	• Final version of Position Manager DFPC: ability to fuse and manage the outputs of all the localisation DFPCs

Schedule	**Visualiser features or capabilities**
M18	• Offline visualization example of application with small subset of ESROCOS types (e.g. pointcoulds and transformation)
M20	• Offline visualization for the sensor data types (ESROCOS types) that OG4 provides.
M22	• Offline visualization of logged data for all InFuse data products.

Schedule	**Data flow control features or capabilities**
M18	• Offline replay of the execution of an example DFCP based on logged data from this same DFPC.
M20	• Automatic reconstruction of any implemented DFPC for offline replay from its description file.
M22	• Offline replay of the execution of any DFCP based on logged data. • Replacement of DFNs from the original DFPC to alternative ones for evaluation.

Schedule	**Filtering and outlier removal features or capabilities**
M18	• Identify use cases for outlier removal • Implement prototypes of outlier removal methods
M20	• Implement outlier removal methods in C++ • Integration of C++ components in pySPACE
M22	• Evaluate outlier removal methods with data from robot

Schedule	MW Facilitator features or capabilities
M18	• Automatic generation of the DFN Common Interface and of the DFPC Common Interface
M20	• Generation of the required code for running a DFPC in a Rock RCOS • Experimental code generation of ROS wrappers for DFNs and experimental assembly DFN ROS nodes to create a distributed DFPC (build and launch files to be evaluated)
M22	• Rock RCOS integration of the Orchestrator • Rock RCOS integration of the Central Data Products Manager • Generation of the required code for running a complete Data Fusion Solution for the Planetary Track in a Rock RCOS • Building the generated DFN ROS Nodes and run time testing of interfaces • Testing the ROS node DFNs (and possible configuration of a distributed DFPC) with sensor data from ROS bags and evaluating outputs

4 Internal unitary and integrated testing (pre-TRR)

4.1 Unitary test plan with data flow control

4.1.1 DFNs

Describes the procedure for testing and evaluating each DFN category.

4.1.1.12 DFN: Force Mesh Generator (SPACEAPPS)

Inputs: - Desired end-effector position - End-effector force measurements - Estimated rover pose relative to target.	**Outputs:** - 3D point cloud representing touched shapes
Test procedure: - Run the node with sample data - A ground truth pointcloud will be used.	**Evaluation criteria or metrics:** - The 3D point cloud generated is a subpart of the ground truth pointcloud.

4.1.1.14 DFN: Image Pre-Processing (USTRATH)

Inputs:	**Outputs:**
- Image - Camera Calibration Matrix - Distortion Coefficients - Downsampling Factor	- Downsampled, Calibrated and Undistorted image
Test procedure: - Run the node on desired image with calibration matrix and coefficients as parameters - Evaluate resulting image	**Evaluation criteria or metrics:** - Image should be the correct resolution and qualitatively undistorted and planar as well as visually matching a test image after undistortion and calibration.

4.1.1.15 DFN: Feature Detection (USTRATH)

Inputs:	**Outputs:**
- Image	- List of features from image (Harris or ORB)
Test procedure: - Run the node on desired undistorted and downsampled image - Check that features have been detected	**Evaluation criteria or metrics:** - Number of features should be similar to a comparison case

4.1.1.16 DFN: Feature Matching (USTRATH)

Inputs:	Outputs:
- First Image - Features in First Image - Second Image - Features in Second Image	- List of matches between features (using Brute Force or FLANN)
Test procedure: - Run the node on desired pair of images with features - Check that matches have been detected	**Evaluation criteria or metrics:** - Number of matches should be similar to a comparison case

4.1.1.17 DFN: Fundamental Matrix Calculation (USTRATH)

Inputs:	Outputs:
- Set of matches between two images	- Calculated fundamental matrix describing homography between images
Test procedure: - Run the node on set of matches - Check that fundamental matrix has been calculated	**Evaluation criteria or metrics:** - Matrix should describe valid homography between two images (algorithm is not guaranteed to find a successful homography)

4.1.1.18 DFN: Triangulation (USTRATH)

Inputs:	Outputs:
- Set of matches between two images - Fundamental matrix	- Local point cloud describing triangulated point locations in space

Test procedure:	**Evaluation criteria or metrics:**
- Run the node on set of matches and valid fundamental matrix - Check that point cloud is valid	- Point cloud should approximately match points in 3D space

4.1.1.19 DFN: Point Cloud Assembly (USTRATH)

Inputs:	**Outputs:**
- Local point cloud - Global point cloud	- Global point cloud incorporating new points
Test procedure:	**Evaluation criteria or metrics:**
- Run the node on local and global point cloud - Check that common points have been merged and new points have been added	- Point cloud should approximately match model used for images in 3D space

4.1.1.20 DFN: 3D Keypoint Selection (USTRATH)

Inputs:	**Outputs:**
- Point cloud - Keypoint descriptor radius - Keypoint spacing	- Keypoints and 3D descriptors for point cloud (SHOT or FPFH)
Test procedure:	**Evaluation criteria or metrics:**
- Run the node on point cloud - Check that keypoints have been selected at appropriate radius	- Keypoints should be selected at appropriate locations

4.1.1.21 DFN: 3D Descriptor Matching (USTRATH)

Inputs: - 3D keypoint descriptors for scene - 3D keypoint descriptors for model	**Outputs:** - Matches between descriptors (Brute Force or FLANN)
Test procedure: - Run the node on scene and model - Check that appropriate matches have been made	**Evaluation criteria or metrics:** - Keypoints should be matched with similar keypoints (this algorithm is not guaranteed to find all matches)

4.1.1.22 DFN: 3D Correspondences (USTRATH)

Inputs: - Scene point cloud - Model point cloud - List of matches between model and scene	**Outputs:** - List of potential poses in order of likelihood (number of matches)
Test procedure: - Run the node on scene and model with matches - Check that poses are reasonable	**Evaluation criteria or metrics:** - Poses should follow position of model shape within scene (this algorithm is not guaranteed to find all poses)

4.1.1.23 DFN: Bundle Adjustment (USTRATH)

Inputs: - Existing Target Point Cloud - Matches for features in original input images - Original input 2D images	**Outputs:** - Point cloud with improved geometric consistency

Test procedure:	**Evaluation criteria or metrics:**
- Run the node on point cloud - Check that resulting point cloud has improved geometry	- Similarity of point cloud to actual scene

4.1.1.24 DFN: Pose Filtering (USTRATH)

Inputs:	**Outputs:**
- List of poses from tracking - Movement model	- List of filtered poses
Test procedure:	**Evaluation criteria or metrics:**
- Run the node on list of poses - Check that poses are reasonable	- Poses should be consistent with movement model

4.1.1.10 DFN: Pose Estimator - Prediction (MAG)

Inputs:	**Outputs:**
- Timestamp - Previous camera pose - Previous detected feature positions	- Predicted feature positions
Test procedure:	**Evaluation criteria or metrics:**
- Run the DFN on various initial chaser poses, - Evaluate estimated pose prediction	- Compute the standard deviation between the estimated reprojected feature positions and ground truth.

4.1.1.10 DFN: Pose Estimator - Correction (MAG)

Inputs:	**Outputs:**
- Chaser pose from AHRS	

- Current feature matches (point cloud relative pose or matched feature sets), - Range measurement from radar - EKF state vector	- Estimated chaser relative pose
Test procedure: - Run the DFN on various initial chaser poses, - Compare estimated chaser pose to ground truth.	**Evaluation criteria or metrics:** - Successful tracking over full trajectory, - Standard deviation between the estimated chaser pose and ground truth.

4.1.1.10 DFN: Point Tracking (MAG)

Inputs: - Current detected image feature - Previous set of detected image features - Coarse relative pose estimation between previous and current frames.	**Outputs:** - Matched 2D feature in previous and current image frames
Test procedure: - Run the DFN on a chosen set of detected features and related pose estimation, - Evaluate matching performance.	**Evaluation criteria or metrics:** - Ratio of false positives falls within tolerance. - Matching success ratio.

4.1.1.10 DFN: Stereo Rectification (MAG)

Inputs: - Stereo Image Pair	**Outputs:** - Rectified Stereo Image Pair

Test procedure:	**Evaluation criteria or metrics:**
- Run the node on a stereo image pair - Check that the image planes are aligned (rotation, focal length)	- Compute the standard deviation of the distance between corresponding epipolar lines - Verify that the sum of squared distances is minimized.

4.1.1.10 DFN: Stereo Correlation (MAG)

Inputs:	**Outputs:**
- Rectified Stereo Images Pair	- Image of Filtered Disparities
Test procedure:	**Evaluation criteria or metrics:**
- Run the node on a rectified stereo image pair - Check that the Disparity Image corresponds to the observed environment	- In simulation, compute the standard deviation of the distance between the the ground truth depth and the depth computed by the disparity image

4.1.1.10 DFN: Template Detection (MAG)

Inputs:	**Outputs:**
- Target template - RGB image - Depthmap	- Target template pose
Test procedure:	**Evaluation criteria or metrics:**
- Provide an RGB image in which the template position is known - Check if the computed position is coherent.	- Detection success rate. - The error between the known pose and the computed one is lower than a specific threshold.

4.1.1.10 DFN: Template Tracking (MAG)

Inputs:	**Outputs:**
- RGB Image - Initialization pose	- Target pose
Test procedure: - Run node - Check during movement that the target is detected and its relative position between target and camera is coherent.	**Evaluation criteria or metrics:** - Ratio of successful tracking over trajectory. - Error between relative pose estimation and ground truth. - Robustness of detection and tracking to various operation conditions.

4.1.1.10 DFN: Point Cloud Registration (MAG)

Inputs:	**Outputs:**
- Initial pose estimation - 2 input point clouds	- Transformation minimizing the distance between the two input clouds
Test procedure: - Provide two point clouds with known transformation and the initial pose estimation, - Check if the computed transformation corresponds to the ground truth.	**Evaluation criteria or metrics:** - The distance between the computed and the expected transformation is lower than a chosen threshold.

4.1.2 DFPCs (with local DPM)

In this section we describe the testing procedure for each DFPC applicable to the OT-RI associated to use cases defined in D5.1 Section 2.2.

A DFPC test report will be produced after each test DFPC testing session, reporting on: the protocol followed, DFNs involved, expected results vs. obtained results, ref and location (on server) of the log of the tests, and any remarks and recommendations in relation to the results.

4.1.2.1 DFPC: Mid- and close-range visual tracking

Inputs:	**Outputs:**
- Left and/or right images with associated metadata - Initial pose and velocity	- Estimated target pose with respect to the chaser - Estimated local velocity
Test procedure: - Launch DFPC controller - Set DFPC parameters associated to its DFN - - DFPC controller reads sensor data samples	**Evaluation criteria or metrics:** - Pose deviation from ground truth, 30mm in close range, and 5% in mid-range - Data rate up to 3Hz - Tracking success rate, under nominal, over- and under-illumination

4.1.2.2 DFPC: 3D Model Detection and Tracking

Inputs:	**Outputs:**
- Left and/or right images with associated metadata - Estimated pose relative to target.	- Estimated target pose with respect to the chaser - Target object ID
Test procedure: - Launch DFPC controller - Set DFPC parameters associated to its DFN	**Evaluation criteria or metrics:** - Intermediate background removal image present - Filtering image presenting higher PSNR than input - Depth normalisation present

- DFPC controller reads sensor data samples containing known target speed and ID.	- World transformation of points correct with respect to world reference frames - Binning voxelization correctly done, tested via merging occurs - K-means outputting K regions - SURF generated features - Edge detection Image present and similar to correct Edge detection Image - Weighting and Fitting algorithm changing output decision based on parameters

4.1.2.3 DFPC: Far-range Target Tracking

Inputs: - RGB images + metadata - Chaser AHRS data + metadata	**Outputs:** - Estimated chaser pose wrt to target.
Test procedure: - Precisely measure the fixed target position and orientation wrt to global frame. - Teleoperate the chaser on a predefined trajectory while the target is kept within the sensors field of view and the tracking DFPC is activated.	**Evaluation criteria or metrics:** - Error between estimated pose and true chaser relative pose. - Tracking accuracy of the center of the target is in the order of magnitude of 1 pixel on the sensor, which would be sufficient to allow for further rendezvous operations. - Update rate is expected to be sufficiently fast to perform autonomous navigation at 1Hz.

4.1.2.4 DFPC: Mid- and Close-range Detection and Tracking

Inputs:	Outputs:
- Left and Right images + metadata - Depth images + metadata - RGB images + metadata - Chaser AHRS data	- Target detection success/failure - Estimated rover pose relative to target
Test procedure: - Precisely measure the fixed target position and orientation wrt to test field global frame. - Operate the chaser in the vicinity of the target on a given trajectory while the DFPC is running, keeping the target in the FOV of the sensors. - Check resulting poses for accuracy.	**Evaluation criteria or metrics:** - Detection success rate over trajectory higher than 80%, - Error between coarse pose estimation and true chaser relative pose, computed ground truth lower than 15° (angular) and 0.5m (linear), - Final position error is lower than 10cm at 10m range, angular accuracy should be lower than 10° in heading, - Computational performance of algorithm (pose update rate) is compatible with the 1Hz update target.

4.1.2.5 DFPC: LIDAR-based Tracking

Inputs:	Outputs:
- Chaser attitude from AHRS + metadata - Point cloud + metadata	- Estimated chaser pose with regard to target.
Test procedure: - Precisely measure the target position and orientation wrt to global frame. - Load an existing 3D point cloud model of the target.	**Evaluation criteria or metrics:** - Final relative position accuracy should be lower than 0.1m, and angular accuracy should be lower than 10° in heading.

- Operate the chaser in the vicinity of the target on a given trajectory while the DFPC is running, keeping the target in the FOV of the sensors. - Check resulting poses for accuracy.	- Computational performance of algorithm (pose update rate) should be compatible with 1Hz update rate target.

4.1.2.6 DFPC: Haptic scanning

Inputs: - Desired end effector position - End-effector force measures - (optional) Estimated pose relative to target.	**Outputs:** - 3D point cloud representing touched shapes
Test procedure: - Launch DFPC controller - Set DFPC parameters associated to its DFN - DFPC controller reads sensor data samples	**Evaluation criteria or metrics:** - Octomap are correctly generated - Mesh generator successfully generates data - Force field correctly merged - For each collision with model, a force measure has been recorded and associated with current point

4.1.2.7 DFPC: Point Cloud Reconstruction

Inputs: - Left and/or right images with associated metadata - Initial Pose Estimation	**Outputs:** - 3D point cloud representing close-range objects in scene - Estimated pose of camera
Test procedure: - Launch DFPC controller - Set DFPC parameters associated to its DFN	**Evaluation criteria or metrics:** - Point cloud is a geometrically accurate representation of scene

- DFPC controller reads point cloud of scene and sensor pose	- Pose of sensor is consistent with actual sensor pose from images

4.2 Integrated test plan with data flow control

Using the CDFF-Dev features, the data fusion solutions will be tested offline, once log data correspondent to the mission is collected.

4.2.1 DFPCs + Orchestrator

Validate the correct functioning of the orchestrator using real data and DFPCs offline. The data to be used for these test should be similar to the ones that will be obtained in the final scenarios. When possible acquired from the same sensors.

It will be possible to analyze the capabilities of the orchestrator using different configurations of the orchestrator with a variety of DFPCs.

Test approach:

The user will provide the path to the logged data in a Python script. In this python script the orchestrator will be accessible (thanks to Python bindings) as well as the DFPCs. The script will provided the data to the different DFPCs and its correspondent DFNs. The Orchestrator based on the results of the processed data and on the incoming data itself, can decide to stop or activate different DFPCs. This activation and deactivation of DFPCs will be implemented in the python side by querying the orchestrator which DPFCs should be activated on a frequency that will be defined by the developer.

Pass criteria: The test will be considered as successful if the result of the DFPC orchestration is sound and allows the proper generation of DFPC products, that typically would meet OG2 / ERGO needs.

4.2.2 DFPCs + DPM (local & central DPM)

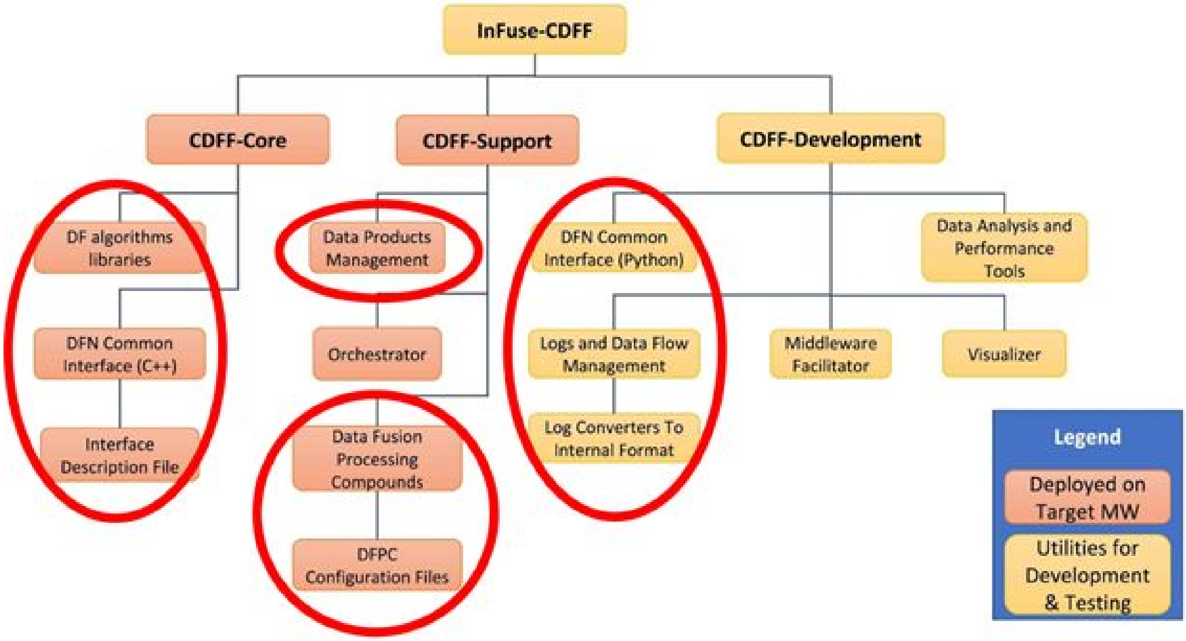

Image 3: CDFF product tree: focus on DFPC + DPM + data flow

In this test, the focus is on the interactions between the DPM and the DFPCs. Such testing activities require the Data Flow component of the CDFF, so that to conveniently and effectively transfer various types of data between the DFNs involved in DFPCs to be tested, and the DPM components.

Also, the Data Analysis and Performance Tools (aka. DAP Tools) will optionally be used as part of the tests, so that to (1) verify the proper functioning of these tools and (2) obtain relevant information on the performances of the data fusion algorithms involved in the tests.

The test will consist of:

(1) Running a single DFPC, and verifying that relevant data products are properly received and handled by the DPM.
(2) Running a single DFPC, and verifying that it can effectively retrieve and access data products it needs, from where it is stored in the DPM.
(3) Running several (at least 2) DFPCs in parallel, and ensuring that data products transfer from a DFPC to another, through the DPM, works properly.

Pass criteria: The test will be considered as successful if it is verified that the DPM can properly handle (i.e. receive and store) the data products that need to be shared among the DFPC, and can suitably make data products available to DFPCs, on request.

The global DPM can be evaluated through its common interface in the data flow control environment. Local DPMs will as well be supported, for this the python bindings to the local DPMs will have to be provided by the developers.

An example of a test that will be performed is the sharing of data through the central DPM between two DFPCs: One of the DFPCs will store a map in the DPM and a second DFPC will query and use internally this map.

A python script where this test is performed will be provided.

4.2.3 DFPCs + DPM (centralized) + Orchestrator

This test deals with a fully integrated CDFF, with the purpose of ensuring overall consistency of data exchanges and process triggered within each component. No mock components are required in this setup though.

A baseline test scenario would consist in the following steps:

(1) The Orchestrator selects and enables a DPFC, as a reaction to an assumed request from OG2 (will in the test be triggered directly at the Orchestrator level).

(2) The selected DPFC is enabled, waiting for inputs from sensors.

(3) In the absence of OG4 sensors, pre-recorded data sets are fed (“manually”) in the CDFF, through the Data Flow.

(4) The DFPC should be able to receive and process the input data, and shall produce and dispatch (through the Data Flow) a sound data fusion product.

(5) The data product is sent to the DPM, and the Orchestrator is notified of its availability.

(6) The Orchestrator requests the resulting data product from the DPM.

(7) The DPM passes the data product material to the Orchestrator.

During all the process, the Data Visualizer is able to render relevant data samples and data products, on request. Similarly the DAP tools may be used in support to performances shaping.

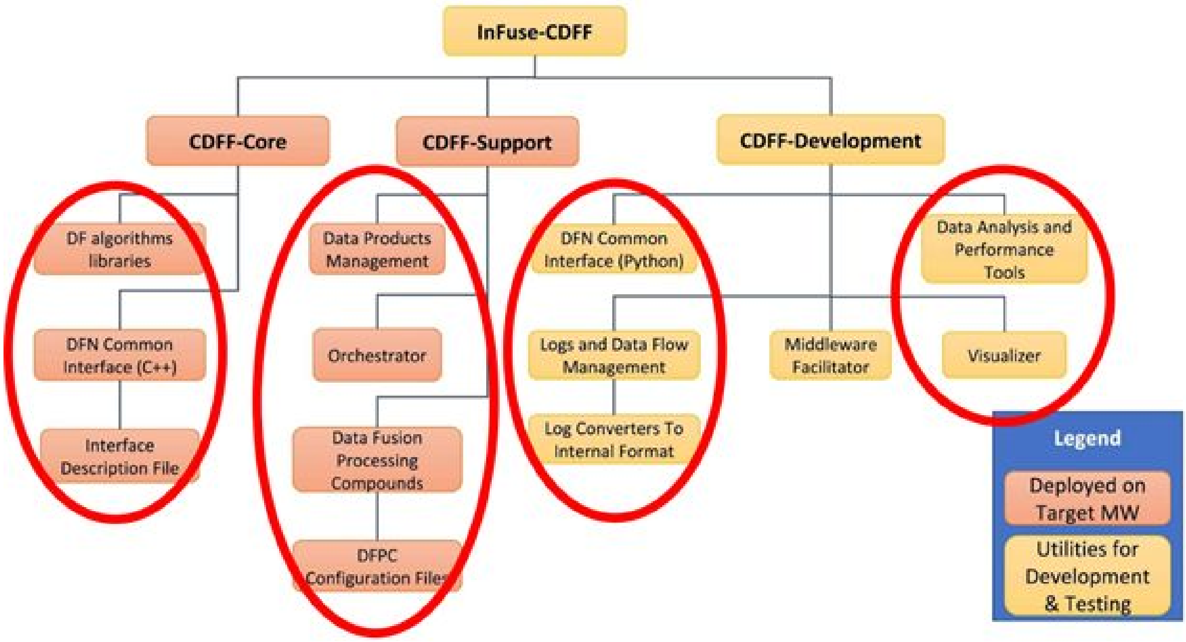

Image 4: CDFF product tree: All integrated

Pass criteria: The test will be considered as successful if it can be verified that the integrated CDFF components behave nominally, and that multiple (and representative) DFPCs can be successfully solicited during the execution of scenarios comparable to the baseline test scenario above. The central DPM would play a role in allowing DFPCs to exchange historic and current pose estimates generated from other DFPCs.

4.2.4 Orchestrator with OG2 (data+command) and OG4 (command) mock interfaces

For what concerns the interface between the Orchestrator and OG2, the interfaces concerned are essentially the ones allowing OG2 to request data products to OG3, and allowing OG3 to return requested data products.

For what concerns the interface between the Orchestrator and OG4, the interfaces concerned are those allowing to set the operational mode of OG4, i.e. selecting one of the working modes for the sensors.

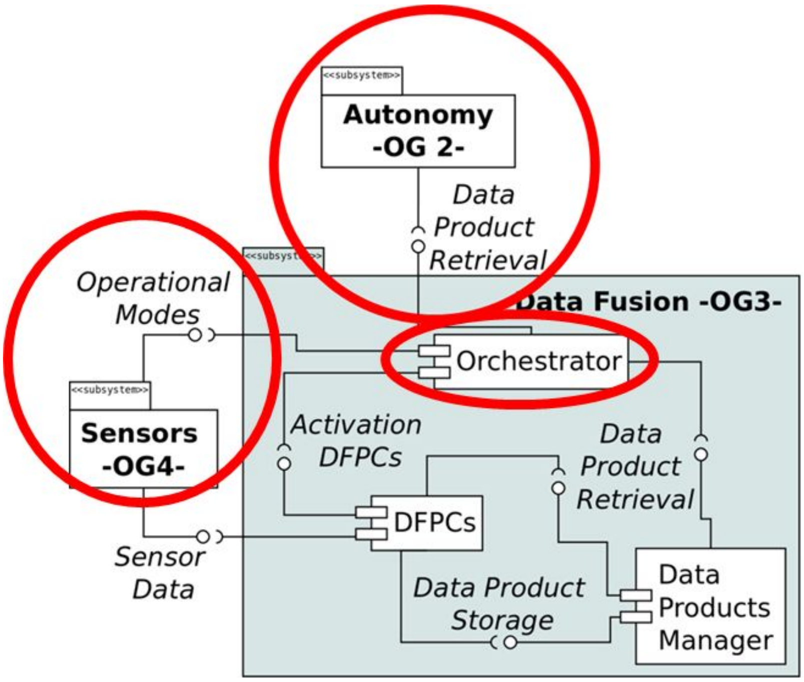

Figure 5: Orchestrator interactions with OG2 and OG4.

Test approach:

In order to test the identified interfaces, a “mock OG2” software (aka. M-OG2) and “OG4 emulation software” (aka. E-OG4) will be purposely developed respectively as a mean to issue requests towards the Orchestrator and to ensure that received data products match expectations (structure, content) in the case of OG2 interfaces, and as a mean to receive and verify the structure and content of mode setting requests, for concerns the OG4 interfaces. The set of interfaces that will be tested between OG3, OG2 and OG4 are in the Appendix 7.2.

In order to provide the Orchestrator with relevant data products, either a mock DPM or real DPM will be used to provide the Orchestrator with the required data products (as pre-recorded samples) – through the CDFF Data Flow, for the sake of convenience.

These interfaces tests will be carried out by wrapping the in Orchestrator’s OG2 and OG4 related APIs in TASTE environment, a common RCOS between OG4 and OG3 or directly linking interfaces between them, and the data products will be expressed in (extended) ASN.1 formalism as defined by OG1.

Pass criteria: The test will be considered as successful if it can be verified that the Orchestrator:

- Can handle requests issued by M-OG2 for all relevant Data Products that OG2 may possibly request, and accordingly each time returns to the M-OG2 software the

expected Data Products in the desired data format (Ex. ASN1.0) and quality (resolution, respecting error margins..).
- Can properly send operation mode setting requests to E-OG4, i.e. as a data structure and content that E-OG4 will validate as correct.

4.2.5 DFPC with OG4 mock sensor data interfaces

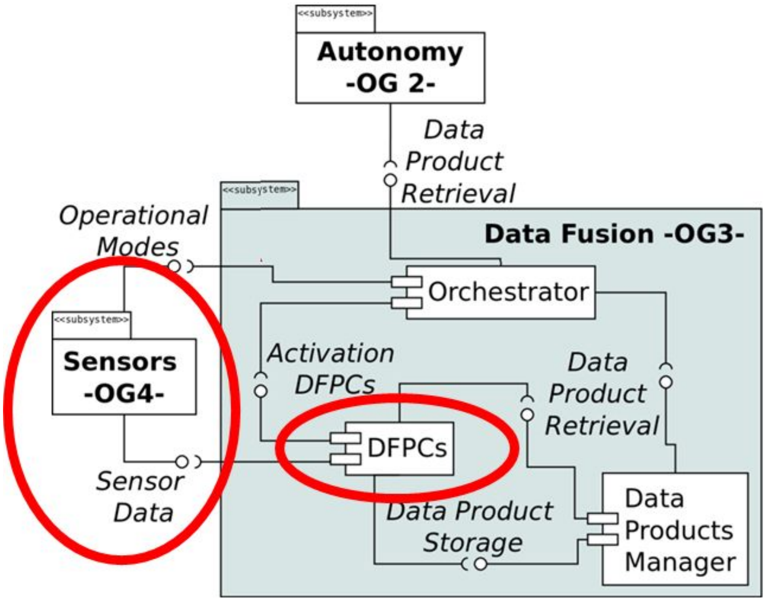

Image 6: DFPCs interactions with OG4

The interface concerned is the one allowing DFPCs to access sensor data produced and provided by OG4.

Test approach:

In order to test that, OG4 emulator software (aka. E-OG4) will be purposely developed as a mean to provide sensors data (whose source may be either pre-recorded samples, simulated, or produced online by real sensors) to the data flow of the CDFF, so that to make these data available to the DFPC

These interfaces tests will be carried out in TASTE environment, a common RCOS between OG4 and OG3 or directly linking interfaces between OG3-OG4. The data issued by E-OG4 will be expressed in (extended) ASN.1 formalism as defined by OG1. OG4 cooperation will be required for defining and developing the emulated sensor data interfaces. The set of interfaces that will be tested between OG3 and OG4 are in the Appendix 7.2.

To test the different types of sensor data interfaces (Ex. IMU, Cameras, Lidar, stereo cameras etc.), will require a few sample DFPCs (one or more) to be wrapped in TASTE, a common RCOS between OG4 and OG3 or directly linking interfaces to acquire data from OG4 emulated sensor data interfaces.

Pass criteria: The test will be considered as successful if it can be verified that the DFPCs are able to access and use efficiently the sensors data incoming from E-OG4. The test coverage should make provision for all sensors and data types foreseen as part of this RI.

4.3 Test schedule with EGSE M18, M20, M22

4.3.1 OOS-Sim

Schedule	Test approach
M18	- Objective is to verify readiness of the facility and initial test of DFPCs - Fully integrate mechanical and electrical subsystems of the OOS-sim - Sun simulator and environment simulation should be ready - Computational hardware should have all the sensor interfaces (e.g multiple ethernet cards) in a specified OS (OSL 42.1) - Cameras should be integrated and image data should be arriving correctly timestamped - IMU should be integrated and data should be arriving correctly timestamped - LIDAR should be integrated and point clouds should be correctly timestamped - CDFF and its dependencies should be deployed on OOS-sim - Perform functional tests of above-mentioned capabilities - Allocate 7 days for calibration and internal testing
M20	- Extensive testing of the Model-based tracker on OOS-sim - Data recording at various illumination and trajectories - Online demonstration
M22	- Full online demonstration of InFuse CDFF at OOS-sim - Release of recorded data
M24*	- Post-processing mid-range data recorded by OG4 using GMV-OG6 facility (DFPCs: detection, reconstruction and tracking) (offline) - Test DFPCs in mid-range with the data recorded by OG4 using GMV-OG6 facility (offline)

*OG4 mid-range data is available latest in M24

5 Testing and validation with OG6 facilities (post-TRR)

5.1 Preparing and verifying OOS-Sim EGSE and interfaces

5.1.1 Sensors and Calibration

For the validation of the orbital RI described in D5.1, it is required to have at least a calibrated stereo camera pair and a LIDAR sensor. The OOS-Sim has currently operational prosilica GC1600H stereo cameras mounted on the robotic manipulator and integrated into the test facility.

The cameras must be connected to a capture device or directly to the InFuse computer that supports simultaneous image capture for both left and right images. The mounting must be fixed such that the distance and angle of both cameras is known as well as the camera calibration coefficients. The two cameras at OOS-sim is currently separated by 6 cm for close-range operation. Ethernet cameras will be powered by 12V Power over Ethernet injectors, and will use Ethernet and the DLR Sensornet API for capture.

The LIDAR will also be connected via Ethernet and provide three dimensional point cloud distance measurements to the InFuse computer. The Mechanical and electrical installation work is currently ongoing. In order to use the test facility for validation of RI, the system should be calibrated. The objective of the calibration is to produce accurate ground truth pose trajectories from robot kinematics. The calibration includes

- Robot kinematic calibration
- Robot-sensor calibration (hand-eye calibration)
- Camera calibration.

The kinematics is already calibrated and ready to use. The hand-eye and a camera calibration will be performed before the actual test. For this purpose, a number of images of a known calibration pattern will be used to estimate internal parameters of a camera and relative transformations by placing the robot arm at a number of viewpoints. For this purpose, the DLR calibration tool CalDe and CalLab will be used.

Verification steps to check if the setup is ready for testing:

- Verify that sensor data is as expected, e.g by using visualization
- Assess accuracy of sensor and hand-eye calibration, e.g by reprojection error

5.1.2 Software and Data interfaces with OBC

The OOS-Sim is equipped with a linux (Opensuse Leap 42.3) computer dedicated to sensor data acquisition and processing. Currently it hosts two ethernet cameras. InFuse will run on the computer next to the data acquisition software.

The sensor (camera, IMU) software driver and data acquisition will be handled with the DLR proprietary software (SensorNet). On the other hand, the DLR middleware (Links and Nodes) will be used, example, for inter-process communication among various nodes such sensor, robot controller. The data rate of sensors should be specified to effectively communicate between various computers. With regard to the DLR OO-sim, the data acquisition rate of the stereo cameras is 25Hz. A Velodyne LIDAR VL-16 scans 300,000 points per sec, whereas the Xsen MTI IMU update rate is 120 Hz. However, the processing rate depends on the algorithm employed.

5.1.3 Ground truth

The ground truth refers to the relative transformation from the sensor coordinate to the target frame. This transformation is obtained from transformation chain of the robot joints and sensor-robot (hand-eye) calibration. Hence, the accuracy of the ground truth relies on the accuracy of the hand-eye and robot calibration. In order to compensate some offset in robot joints, an external dedicated active marker based tracker is employed. The evaluation of the CDFF requires recording of a synchronized sensor data and robot motion. The recording of the ground truth data can be performed after the system is fully calibrated.

We separate the recording of data and demonstration depending on the type of testing (offline/online).

- For the offline validation, CDFF will not be deployed on the test facility during recording of data. The CDFF will be rather validated on the recorded data offline.
- For the online validation, the CDFF will be deployed on the test facility and ground truth and estimated poses or points will be logged during operation. However, image sequences and point clouds will not be recorded during live demonstration as they slow the system.

The result of pose estimation is compared with the ground truth pose in each axis and will be shown graphically as well as statiscically. Currently, the accuracy of the groundtruth of the DLR OOS-sim <5 mm and <0.5 deg in translation and rotation respectively, whereas that of GMV is order of 0.1 mm.

5.1.4 Datasets and Lighting Condition

Datasets will be generated by selecting typical trajectories from on-orbit servicing scenarios. For this purpose, motion trajectories can be produced from the target and servicer. Two cases of target satellite will be considered:

- attitude controlled satellite, where on-board attitude measurement sensors are functional, but with small attitude disturbance. Here, the problem of pose estimation is reduced to estimation of position, as the orientation measurement can be obtained by actively aligning servicer and target satellite with an on-board GNC subsystem.
- Non-cooperative target, where the target is tumbling or spinning with angular velocity (typically, max 4 deg/s) a very small linear velocity. In this case, both position and orientation of the target have to be estimated.

On the other hand, the servicer motion can be controlled with its fully functional GNC system commanded to the desired motion profile. We consider two motion profiles of the servicer:

pure translation and roto-translation with a controlled velocity. Therefore, the motion trajectories for the validation of DFPC will be generated from the relative motions of the target and a servicer as described above.

On the other hand, it is essential that the DFPCs will be validated under space lighting condition. The space lighting is simulated with high power floodlight, which can be placed in a limited direction with respect to target because of the space limitation. We choose a typical direction of light ranging from nominal illumination to under and over illumination depending on the sun direction with respecto the target, denote as P1, P2, P3 and P4 in Fig.7 below.

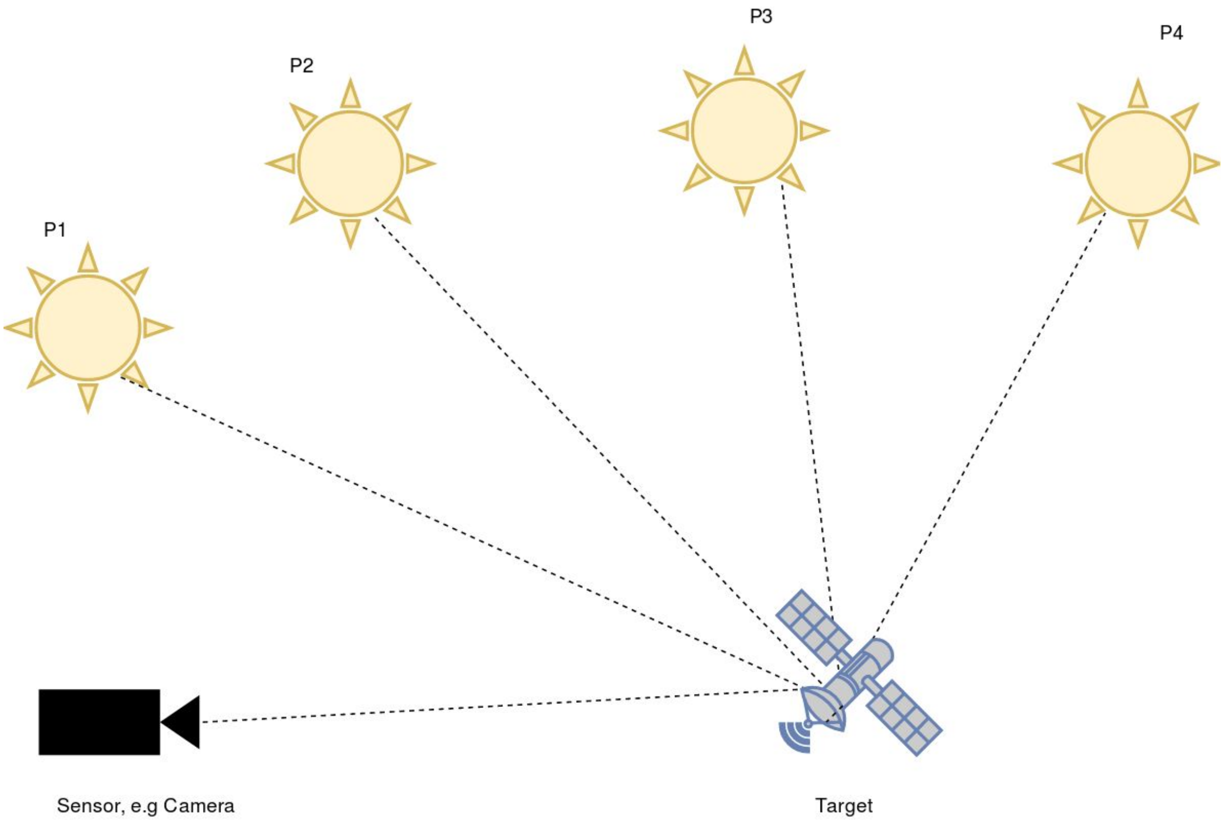

Image 7: The sun direction with respect to the target.

The validation of the DFPC consist in possibly 5 sun directions (Table below), which includes important illumination conditions that has to be tested.

Hence, we generate datasets by combining these lighting positions with angular velocity of the target, typically from 0 deg/s to 4 deg/s in steps of 1 deg/s . Thus, we will have at least 20 various test datasets for the validation of DFPCs. Based on this motion profile and lighting condition, data analysis will be performed and technical results and lessons learned will be disseminated.

Sun position	Angle with respect to sun [deg]	Lighting condition
P1	30	over-illumination
P2	60	nominal
P3	90	under-illumination
P4	120	Near eclipse

5.1.5 Validation schedule

The validation should be performed primarily at the OOS-sim, followed by the offline validation with the datasets recorded by OG4 at the GMV-OG6 facility available in M24. The OG6 facility (OOS-sim) will be available for validation and verification of CDFF for the first test in M18. As we follow a continuous integration approach in InFuse software development, at least one DFPC, for example model-based tracking will be validated in M20. A further and a consolidated validation will be performed in M22. The duration of the test at OOS-sim will be one week assuming a calibrated sensors. Partners who provide their DFPC will be present to support the integration and execution during validation.

Detailed Steps for validating the scenario are provided in the table below.

Day #	**Validation Steps**
1	Setup of sensors on the OOS-Sim Connecting OBC and network interfaces to sensors
2	Calibrating sensors Basic test coverage, ensuring the entire setup properly works
3	Open loop control of the set up by OG6 - pre-planned maneuvers will be tested Close-range model-based tracking - open loop and online

4	Close-range model-based detection- open loop and offline
5	Potential list of DFPCs that will be tested - in open loop offline Mid-range tests on recoded data at GMV-OG6 facility by OG4: - Model based detection - Model based tracking - LIDAR based 3D tracking - Reconstruction
6	Final demo

6 Conclusion

This deliverable documents the unitary and integrated test plans for the Orbital RI of the InFuse CDFF.

Besides describing the overall approach, the release schedule of the core components of InFuse is provided as a best estimate at the time of issuing this deliverable.

The release schedule is synchronized with internal integration and testing rounds, that will take place every 2 months during the implementation phase.

The produced source code will be continuously tested and verified, relying on a common git framework with proper automatic build routines. This will ensure that the overall CDFF software is all the time sound while components are progressively added to it.

Finally this deliverable also documents the testing and validation foreseen with OG6 facilities, during the post-TRR activities.

This document should serve as a reference for testing and integration activities of InFuse, but the content may likely be the object of adjustments during the implementation phase, due the possibilities for contingency (e.g. adapting the release timing or order for some of the components, or adapting the testing procedures).

The Test Readiness Review should anyway refer to the test plan proposed in this document (or its latest version, should changes be recorded).

7 Appendix

7.1 CDR Requirements Traceability Matrix

The requirements were organized in the System Requirements Document (deliverable 3.2) according to a previous categorization of the features. In this appendix the requirements remain organized as previously to facilitate tracking with the previous document.

7.1.1 CDFF Core

7.1.1.1 User Requirements (UserR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_UserR_A101	Yes	4.1.1 (4.1.1.7, 4.1.1.8, 4.1.1.9, 4.1.1.15, 4.1.1.16) 4.1.2.2, 4.1.2.4	
SR_UserR_A102			

	Yes	4.1.1, 4.1.2.1, 4.1.2.3, 4.1.2.5	
SR_UserR_A103	Yes	4.1.1, 4.1.2.1, 4.1.2.5, 4.1.2.7	
SR_UserR_A104	Yes	4.1.1, 4.1.2	
SR_UserR_A105	Yes	4.1.1, 4.1.2	

SR_UserR_A106	Yes	4.1.1, 4.1.2	
SR_UserR_A107	Yes	4.1.1, 4.1.2	
SR_UserR_A108	Yes	4.1.1, 4.1.2	
SR_UserR_A109	Yes	4.1.1, 4.1.2	

SR_UserR_A110	Yes	4.1.1	
SR_UserR_A111	Partial	5	This will be tested in OG6 facilities with regular computing platforms with imposed memory and processing constraints
SR_UserR_A112	Yes	4.2, 4.2.4	
SR_UserR_A113	N/A		DFNs in 4.1.1 are already selected as a result of the trade-off

SR_UserR_A114	Yes	4.1.1, 4.1.2	Each DFN& DFPC has a test plan associated with it to meet RAMS
SR_UserR_A115	Yes	5.1	
SR_UserR_A116	Partial	5.1	Visualization is only provided on the Developer's Environment
SR_UserR_A117	N/A		Covered in D5.3

7.1.1.2 Functional Requirements (FuncR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_FuncR_A201	Yes	4.1.2.3	
SR_FuncR_A202	Yes	4.1.2.1, 4.1.2.2, 4.1.2.3, 4.1.2.4, 4.1.2.5	
SR_FuncR_A203	Yes	4.1.2.1	

SR_FuncR_A204	Yes	4.1.2.7	
SR_FuncR_A205	Yes	4.1.2.1, 4.1.2.2, 4.1.2.4, 4.1.2.5	
SR_FuncR_A206	Yes	4.1.2.1, 4.1.2.2, 4.1.2.4, 4.1.2.5	
SR_FuncR_A207	N/A		Planetary track

SR_FuncR_A208	N/A		Planetary track
SR_FuncR_A209	N/A		Planetary track
SR_FuncR_A210	N/A		Planetary track
	N/A		Planetary

SR_FuncR_A211			track
SR_FuncR_A212	N/A		Planetary track
SR_FuncR_A213	N/A		Planetary track
SR_FuncR_A214	N/A		Planetary track
		4.1.2.1,	

SR_FuncR_A215	Yes	4.1.2.2, 4.1.2.3	
SR_FuncR_A216	Yes	4.1.2.5	
SR_FuncR_A217	Yes	4.1.2.1, 4.1.2.2, 4.1.2.3	
SR_FuncR_A218	Yes	4.1.2.5	

SR_FuncR_A219	Yes	4.1.2.5	
SR_FuncR_A220	Yes	4.1.2	
SR_FuncR_A221	Yes	4.1.2.3	
SR_FuncR_A222	Yes	4.1.2.6	
		4.1.1,	

SR_FuncR_A223	Yes	4.1.2	
SR_FuncR_A224	Yes	4.1.1, 4.1.2	
SR_FuncR_A225	Yes	4.1.1, 4.1.2	
SR_FuncR_A226	Yes	4.1.1, 4.1.2	

SR_FuncR_A227	Yes	4.2.5	
SR_FuncR_A228	N/A		Planetary track

7.1.1.3 Performance Requirements (PerfR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**
SR_PerfR_A301	Yes	4.1.2, 5.1.3	

SR_PerfR_A302	N/A		Planetary track
SR_PerfR_A303	N/A		Planetary track
SR_PerfR_A304	Yes	4.1.2.3	
SR_PerfR_A305	Yes	4.1.2.4	

SR_PerfR_A306	Yes	4.1.2.4, 4.1.2.6	
SR_PerfR_A307	Yes	4.1.2.1, 4.1.2.3, 4.1.2.4	
SR_PerfR_A308	Yes	4.1.2, 5.1.3	

7.1.1.4 Interface Requirements (IntR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**

SR_IntR_A403	Yes	4.2.2, 4.2.3, 4.2.4, 4.2.5	
SR_IntR_A404	Yes	4.2.2, 4.2.3, 4.2.4, 4.2.5	
SR_IntR_A405	Yes	4.2.2, 4.2.4	
SR_IntR_A406	Yes	4.2.2, 4.2.4	

SR_IntR_A407	Yes	4.2.2, 4.2.4	
SR_IntR_A408	Yes	4.2.2, 4.2.4	
SR_IntR_A409	Yes	4.2.2, 4.2.3	

7.1.1.5 Resource Requirements (ResR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ResR_A501	Yes	5.1, 4.1.2	
SR_ResR_A502	Yes	5.1.2	

SR_ResR_A503	Yes	4.1	
SR_ResR_A504	No	-	PROM usage is not known in this phase.
SR_ResR_A505	Yes	5.1.2	

7.1.1.6 Operational Requirements (OpR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**

SR_OpR_A601	Partial	4.2.1, 5.1	Visualization is only provided on the Developer's Environment
SR_OpR_A602	Partial	4.2.1, 5.1	Visualization is only provided on the Developer's Environment
SR_OpR_A603	Yes	4.2.1	Orchestrator core functionality
SR_OpR_A604	Partial	4.1.2	Possible in the developers environment

7.1.1.7 Product assurance and safety requirements (ProdR)

Not Applicable.

7.1.1.8 Configuration and implementation requirements (ConfR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments
SR_ConfR_A801	Yes	4.2.2, 4.2.3	
SR_ConfR_A802	Yes	4.2.2, 4.2.3, 4.2.5	
SR_ConfR_A803	Yes	4.1	All DFPCs and DFNs are coded as per InFuse coding guidelines making it compatible

			for space grade architectures.
SR_ConfR_A804	Yes	4.1	All DFPCs and DFNs are coded as per InFuse coding guidelines making it compatiblefor space grade architectures.
SR_ConfR_A805	Yes	5.1.2	RTEMS is in D5.3
SR_ConfR_A806	Yes	5.1.2	Architecture described in D5.3
SR_ConfR_A807	Yes	5.1.2	Architecture described in D5.3

7.1.1.9 Test and Validation (ValR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**
SR_ValR_A901	Yes	5.1.3	
SR_ValR_A902	Yes	5.1.3	

SR_ValR_A903	Yes	5.1	
SR_ValR_A921	N/A		Planetary Track
SR_ValR_A922	N/A		Planetary Track
SR_ValR_A923	N/A		Planetary Track

SR_ValR_A924	N/A		Planetary Track
SR_ValR_A951	Yes	5.1.4	
SR_ValR_A961	N/A		Planetary Track

7.1.2 CDFF Dev

7.1.2.1 User Requirements (UserR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments

SR_UserR_B101	Yes	2.1, 4.2	
SR_UserR_B102	Yes	2.1, 4.2.4, 4.2.5	
SR_UserR_B103	Yes	4.2.2, 4.2.3	
SR_UserR_B104	Yes	4.2	Visualization is providedin CDFF-Dev tools

SR_UserR_B105	Yes	2.1	Provided in CDFF-Dev tools
SR_UserR_B106	Yes	4.2	
SR_UserR_B107	Yes	4.2	

7.1.2.2 Functional Requirements (FuncR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**

SR_FuncR_B201	Yes	2.4, 5.1.4	
SR_FuncR_B202	Yes	4.2.2, 4.2.3	
SR_FuncR_B203	Yes	4.2	Visualization is provided in CDFF-Dev tools
SR_FuncR_B204	Yes	2.1	Provided in CDFF-Dev tools
	Yes	4.2.1	

SR_FuncR_B205			
SR_FuncR_B206	Yes	4.2.1	
SR_FuncR_B207	Yes	4.2.1	
SR_FuncR_B208	Yes	4.2.1	
			Visualization is provided in CDFF-Dev tools

SR_FuncR_B209	Yes	4.2	
SR_FuncR_B210	Yes	4.2.2, 4.2.3	
SR_FuncR_B211	Yes	4.2.1	

7.1.2.3 Performance Requirements (PerfR)

NA.

7.1.2.4 Interface Requirements (IntR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**

SR_IntR_B401	Yes	4.2.1, 4.2.4	

7.1.2.5 Resource Requirements (ResR)

Req. ID	**Compliance (yes / no / partial)**	**Section where it is addressed**	**Comments**
SR_ResR_B501	Yes	2.1, 2.2	
SR_ResR_B502	N/A		Depends on DFPC and is addressed in D5.3

SR_ResR_B503	Yes	5.1.2	

7.1.2.6 Operational Requirements (OpR)

NA.

7.1.2.7 Product assurance and safety requirements (ProdR)

NA.

7.1.2.8 Configuration and implementation requirements (ConfR)

Req. ID	Compliance (yes / no / partial)	Section where it is addressed	Comments

SR_ConfR_B801	Yes	4.2	

7. 2 Detailed Interface among OGs

Interface	Call Direction	Synch/Asynch	Params	Possible Values	Return value	Comments
setCDFFState	OG2 -> OG3	Synch	State	Initialize, idle, reset, stop	Success, Error, invalid States	
getCDFFState	OG2-> OG3	Synch	NULL	N/A	Runtime state or error	
getDFPCStatus	OG2->OG3	Synch	Type	DEM or Pose	Runtime state or error	
getRoverMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution, area of coverage	sensor names, expected accuracy values, Hz, pixel to cm coverage, in sq. mts	DEM map or error state	Map produced with information gathered by sensors on the rover itself at the last sensing capture
getFusedRoverMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution,	sensor names, expected accuracy values,	DEM map or error state	Map produced with information

			area of coverage	Hz, pixel to cm coverage, in sq. mts		gathered by sensors on the rover itself at the last and previous sensing captures.
getFusedTotalMap	OG2 -> OG3	Synch	List of sensors, accuracy, update rate, resolution, area of coverage	sensor names, expected accuracy values, Hz, pixel to cm coverage, in sq. mts	DEM map or error state	Map produced with information from any sensing sources at any capturing time, e.g. rover, orbital, other mobile or static devices on the surface.
getLocalPose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the LocalPose Pose of the BodyFrame in the LocalTerrainFrame
getGlobalPose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	Produces the GlobalPose Pose of the BodyFrame in the GlobalTerrainFrame
getAbsolutePose	OG2 -> OG3	Synch	frame name, List of	frame string, sensor	Pose + uncertainty	Produces the AbsolutePos

			sensors, accuracy, update rate	names, expected accuracy values, Hz		e Pose of the BodyFrame in the AbsoluteFrame
getTargetRelativePose	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	Pose + uncertainty	relative pose (3 axes position and 3 axes attitude) of the target Body Frame expressed in the chaser Body Frame, with associated uncertainties
getTargetRelativeVelocity	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	twist + uncertainty	relative speed (3 axes translation speeds and 3 axes rotation speeds) of the target Body Frame expressed in the chaser Body Frame, with associated uncertainties
getModelOfTarget	OG2 -> OG3	Synch	frame name, List of sensors, accuracy, update rate	frame string, sensor names, expected accuracy values, Hz	3D Model	This interface produces the 3D model of the target spacecraft.

initDFPC	Orch -> DFPC	Synch	DFPC ID	DFPC Name	Success, Error States	
stopDFPC	Orch -> DFPC	Synch	DFPC ID	DFPC Name	Success, Error States	
getDFPCStatus	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
getDFPCPose	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
getDFPCDEM	Orch -> DFPC	Asynch	DFPC ID, Frequency, Callback function ptr	Name, Hz, Function ptr	N/A	
initICU	OG3 -> OG4	Synch	NULL	N/A	Success, Error States	
setOperatingMode	OG3 -> OG4	Synch	OpModeID	ID Number	Success, Error, invalid States	
selectSensorConfiguration	OG3 -> OG4	Synch	SensorID, ConfigurationID	ID number, ConfigID number	Success, Error, invalid States	
getOpModeSensorStatus	OG3 -> OG4	Synch	OpModeID	ID Number	Runtime or error states	
getStereoCamDepthMap	DFPC -> OG4	Synch	NULL	N/A	Depth map or error state	
getStereoCamDisparityMap	DFPC -> OG4	Synch	NULL	N/A	Disparity Map or error state	
getStereoCamPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getStereoCamImages	DFPC -> OG4	Synch	NULL	N/A	Images or error state	
getToFPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getIMUData	DFPC -> OG4	Synch	NULL	N/A	Linear	

					acceleration & angular velocity or error state	
getLidarPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getLaserScan	DFPC -> OG4	Synch	NULL	N/A	Planar 2D PC or error state	
getRadarScan	DFPC -> OG4	Synch	NULL	N/A	2D or 3D PC or error state	
getHRCameraImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getTIRCameraImage	DFPC -> OG4	Synch	NULL	N/A	Image or error state	
getForceTorque	DFPC -> OG4	Synch	NULL	N/A	Wrench data or error state	
getStructuredLightPointCloud	DFPC -> OG4	Synch	NULL	N/A	Point Cloud or error state	
getStarTrackerOrientation	DFPC -> OG4	Synch	NULL	N/A	Orientation data or error state	