DDR Data Recorder Quick Start Guide

Open Source DDR – Configuring for your Autonomy Code

This open source release of DDR requires configuration. It is not a plug and play solution.

The best course of action when you encounter DDR related errors is to close DDR and run it again. It has startup and exit handlers and cleaners that should recover from most data errors.

The error output will point you to the problem. Files being read and written can sometimes be corrupted (.json files, bag files). The best course of action is to delete the files causing errors (or run rosbag reindex, if prompted).

Removing DDR from the installation directory and using catkin build again is also a good troubleshooting method.

1 THIS IS THE INSTALLATION GUIDE. DO NOT SKIP.

DDR requires:

- Ubuntu 16.04
- ROS Kinetic
- Python 3 plus a few packages below.

DDR will NOT work with Python 2.7. DDR is Python3 based.

Environment Setup

1. **Properly set up a catkin workspace with DDR.**

Catkin is the build tool you will use to build ROS packages. You'll interact with it using catkin_tools

If you do not have catkin tools, please follow instructions here:

https://catkin-tools.readthedocs.io/en/latest/installing.html

This assumes you want to create your catkin workspace in `~/ddr_dr`, but you can create your workspace wherever you like.

Create the directory and initialize the catkin workspace

```
mkdir -p ~/ddr_dr/src
cd ~/ddr_dr
catkin init
```

Bind the catkin workspace to a ROS installation (in this case ROS Kinetic) and configure it.

```
cd ~/ddr_dr
catkin config --extend /opt/ros/kinetic
catkin config --cmake-args -DCMAKE_BUILD_TYPE=RelWithDebInfo
catkin config --merge-devel
```

Setup the workspace to point to your install space

```
catkin config --install
sudo mkdir -p /($YOUR_INSTALL_DIRECTORY)
sudo chown -R username:username /($YOUR_INSTALL_DIRECTORY)
catkin config --install-space /($YOUR_INSTALL_DIRECTORY)
```

2. Install the following packages with 'sudo apt-get install'
   a. python3-pip
   b. python3-yaml
   c. python3-tk
   d. python3 → specifically python version 3.5 - all other versions of python 3 WILL NOT WORK WELL with ROS Kinetic on Ubuntu 16.04
      i. If you are on Ubuntu 18.04, use the latest python.
3. pip install rospkg, catkin_pkg, defusedxml, and netifaces:
   ```
   $sudo pip3 install rospkg catkin_pkg defusedxml netifaces
   ```

   a. If you do not have pip, you MAY try sudo apt-get install'ing but it will probably try to blow up your ROS Kinetic installation.

4. **Warning**: If step #3 tries to UNINSTALL a huge list of ROS related packages, DO NOT PROCEED. You tried apt-get install instead of pip3 install.
   a. Instead of the packages in step #3, you will need to try a variety of:
      i. sudo apt-get install python-rospkg-modules
      ii. sudo apt-get install python-catkin-pkg-modules
   b. If this still tries to uninstall most of ROS, try roslaunching DDR per below guidance.

2 Using DDR

Read through this user guide.

Record ROS data more conveniently than you ever have before!

DDR DR General Use

Do the initial catkin build from the source directory

Source your installation space for ddr

Simply roslaunch DDR DR.

$roslaunch ddr_data_recorder ddrGui.launch

Files of importance:

1. DDR Launch file: ddr.launch
   a. ddr_data_recorder/launch/ddr.launch
   b. This has the rosparams you want to modify.
2. KML: kml.xml
   a. ddr_data_recorder/scripts/dynamic_recording/kml.xml
   b. This contains the list of topics to record and the topicGroup mode they are recorded in.
3. Mode State Bridge
   a. ddr_data_recorder/scripts/api/mode_state_bridge_example.py
   b. This bridge interprets changes from your autonomy code and sends them to our DDR API.
   c. By viewing this file and the kml.xml, one can see the way things work.
   d. You will need to modify this bridge to make DDR work with your autonomy software.
   e. You can remove launch of the bridge from the ddr.launch file and use DDR in standalone mode. It will not change topics being recorded without a publication on /ddr/api or /ddr/event – you can use the scripts in headless_DDR/ directory to manually switch DDR's recording profile according to the groups you have in the kml.xml.
4. Markdown
   a. In the markdown directory (ddr_data_recorder/markdown) you will find several files.
   b. These files are built to read from specific topics with various message types.
   c. To build your own readers, use these example files and add them to the main rosbag reporter files.
   d. More detailed documentation will be released at a later date.

3 Troubleshooting

Errors on start troubleshooting:

1. Ownership of installation directory
2. Unresolved dependencies
3. You've properly cloned DDR and initialized + updated the submodules.
4. Sourcing your installation folder
5. Corrupted files (read the error output, remove those files)

Missing, empty, or incomplete Markdown Files:

1. Run markdown.py on individual capture folders.
    a. python3 /ddr_data_recorder/scripts/markdown/markdown.py /path/to/capture
2. Bags are empty - there is no data in the bags therefore markdown is missing.
3. Bags are missing necessary topics.

Empty Bags:

1. Check your network setup. Rostopic list - do topics appear?
2. Try to echo some of the topics that appear.
3. Make sure you sourced your installation directory in the terminal you launched DDR from.

More Detailed Troubleshooting:

ENVIRONMENT

Ubuntu 16.04 ROS Kinetic (desktop full)
Catkin tools ($ sudo apt-get install python-catkin-tools)

GENERAL USAGE WORKAROUNDS

Force Mode Changes
$. scripts/headless_DDR/ddr_event_pub.sh topicGroup **YOUR-MODE** true false kickstart
Replace your mode with options in the KML – manual, idle, teleop, etc

Restart DDR every so often
DDR has self-correcting startup and cleanup. Recommended if your test is in harsh conditions/weather/temperature.

GENERAL TROUBLESHOOTING

DDR Code Crashing

- **SOURCE, SOURCE, SOURCE....always**
    - source your_workspace/install/setup.bash
    - SOURCE IN EVERY TERMINAL YOU USE, doesn't matter if it's in your .bashrc

    - If you do not source properly, you may be running from your workspace source folder. Those files may not have executable permissions. They gain those permissions during the catkin build process.
- **Do you have the EXACT SAME version of code as the system under test?**
    - If not, your message files may not match. You cannot read information from rostopics sent by the system under test.
- **Did you install all dependencies?** Are you seeing a "something missing" or not found error?
- **Check file permissions.**
    - Do you own the install directory? Meaning, can your user level read, write, and execute files in the installation directory? If not, you can't run the code.
- **Remove the copy of ddr_data_recorder from your install directory. Catkin build again.**
- **If you have no bags generating,:**
    - $. scripts/headless_DDR/ddr_event_pub.sh topicGroup (YOUR MODE FROM KML) true false kickstart

Connection Problems / Empty Bags

- **Can you see the autonomy's topics?**
    - $rostopic list
- **Can you echo topics?**
    - $rostopic echo /test/topic/
        - your test topic should be something that publishes periodically for best test results
- **Small bags or missing data?**
    - rosbag info on bags generated in ~/ddr_bags
        - $ rosbag info XXX.bag
    - How many messages are there for /ddr/andBeyonder?
    - You should have 495-505 messages. It should be very close to 500.
    - If not, your system is unable to record rosbag data. You have insufficient hardware power. This is NOT a DDR problem.
    - Are the topics you expected to record in the bag?
        - If not, your computer is not networked to the system under test properly

Testing Tools

verify_data.py

What this tool is:

A tool to assist and speed up verifying data that comes out of DDR to make sure messages weren't missed.

What this tool is NOT:

A tool to verify that the software that DDR is being used to capture output from is working properly.

Usage:

```
rosrun ddr_data_director verify_data.py directory [directory ...]
```

or

```
python3 /path/to/verify_data.py directory [directory ...]
```

Run with the -h or --help flag for more options:

```
$ rosrun ddr_data_recorder verify_data.py --help
```

automated_testing.py

Run roscore in a terminal.

After DDR is installed, you can run this testing script:

```
$automated_testing.py
```

It will launch DDR and simulate a bunch of stress tests.

You should have a bagfile playing or autonomy code running to simulate a data stream for DDR to record.

You cannot stop the script once you start it. Wait for it to finish. If you stop it before it finishes, you will have to individually hunt down the DDR processes and kill them.

4 DDR DR Situational Usage

4.1.1 Launching DDR DR and GUI separately:

Launch DDR DR

1. roslaunch ddr_data_recorder ddr.launch
2. Open a new terminal on your system.
3. Run:
   rosrun ddr_data_recorder gui.py

4.1.2 To start all of DDR at once:

Run:
roslaunch ddr_data_recorder ddrGui.launch