IDM UID
**9FPTZU**

VERSION CREATED ON / VERSION / STATUS
**07 Mar 2014 / 1.4 / Approved**

EXTERNAL REFERENCE

**Plan**

# Nominal Device Support Software Test Plan (STP)

This document defines the Test Plan for Nominal Device Support (NDS).The audience of this document are the test engineers who will execute the Test Plan and produce the Test Report. In addition, the document is useful as a template for device-specific NDS drivers, or it can be referred from the device-specific NDS drivers if generic test cases are applicable.

| *Approval Process* | | | |
|---|---|---|---|
| | *Name* | *Action* | *Affiliation* |
| *Author* | **Isaev S.** | **07 Mar 2014:signed** | |
| *Co-Authors* | | | |
| *Reviewers* | **Di Maio F.**<br>**Makijarvi P.**<br>**Martin V.** | <br>**19 Mar 2014:recommended**<br>**21 Mar 2014:recommended** | **IO/DG/DIP/CHD/CSD/CDC**<br>**IO/DG/DIP/CHD/CSD/CDC**<br>**IO/DG/DIP/CHD/DIAG** |
| *Approver* | **Simrock S.** | **26 Mar 2014:approved** | **IO/DG/DIP/CHD/CSD/CDC** |
| *Document Security: level 1 (IO unclassified)*<br>*RO: Makijarvi Petri* | | | |
| *Read Access* | **LG: SOPRA extra, AD: ITER, AD: External Collaborators, AD: IO_Director-General, AD: IC_OMPE_WG, AD: Division - Control System Division, AD: Section - CODAC, AD: Auditors, AD: ITER Management Assessor, project administrator, RO, LG: NDS Testing EXT, LG: CODAC team** | | |

| *Change Log* | | | | |
|---|---|---|---|---|
| ***Title (Uid)*** | ***Version*** | ***Latest Status*** | ***Issue Date*** | ***Description of Change*** |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v1_4) | v1.4 | Approved | 07 Mar 2014 | Comments are addressed in this document version:<br><br>1. The list of SRS is updated (Di Maio Franck).<br><br>2. Tests which covers known bugs became a part of general tests part with a references on a bugs (Simrock Stefan)<br><br>3. Reference on NDS-SRS-F-055 - Image compression listed in section requirement is removed (Martin Vincent)<br><br>4. Functionality coverage is estimated and it is 50%. (Martin Vincent)<br><br>5. Estimated time for this test plan execution is estimated 5 hours.<br><br>6. Test's list are extended by the DAQ System tests.<br><br>Comments are NOT addressed in this version of document:<br><br>6. Automatic test’s still uncovered by the STP as far automatic test is a change request.<br><br>7. NDS functions for (DAQ, IAS, Timing, health, etc) are up-to-date and tests for different NDS devices separated |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v1_3) | v1.3 | Revision Required | 27 Nov 2013 | Added tests for state machines.<br><br>TP-NDS-002<br><br>TP-NDS-003<br><br>TP-NDS-004 |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v1_2) | v1.2 | Approved | 27 Sep 2013 | CODAC STP templates was applied. |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v1_1) | v1.1 | Revision Required | 25 Apr 2013 | List of added test:<br><br>3.6.1 Create and run an timing EPICS application [TP-NDS-TST-0001]<br><br>3.6.2 Check timestamping by demand [TP-NDS-TST-0002]<br><br>3.6.3 Check timestamp through interrupt [TP- |

| | | | | |
|---|---|---|---|---|
| | | | | NDS-TST-0003]<br><br>3.6.4 Check handling of reason’s arguments [TP-NDS-TST-0004]<br><br>3.6.5 Check handling of FTE event parameters [TP-NDS-TST-0005]<br><br>3.6.6 Check handling of PULSE event parameters [TP-NDS-TST-0006]<br><br>3.6.7 Check handling of CLOCK event parameters [TP-NDS-TST-0007] |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v1_0) | v1.0 | Signed | 02 Apr 2013 | |
| Nominal Device Support Software Test Plan (STP) (9FPTZU_v0_0) | v0.0 | In Work | 29 Mar 2013 | |

# Table of contents

# 1 INTRODUCTION

## 1.1 Purpose and Scope

This document defines the Test Plan for Nominal Device Support (NDS). The audience of this document are the test engineers who will execute the Test Plan and produce the Test Report.

The audience of this document are the staff responsible for managing development of the system under test and developers.

The test plan covers the following aspects of NDS:

1. Execution of unit tests of all systems units and verification of their results.
2. Test procedures for ensuring all requirements are verified.
3. Test procedures for ensuring previously identified anomalies do not recur (regression tests).
4. Performance tests.

## 1.2 Overview

NDS is a support library for developers of EPICS device support. As such, it is difficult to test independently. Therefore, a "dummy device" implementation has been provided which uses filesystem pipes to mimic inputs and outputs. This device is called the *Pipe Example* and is shown in Figure 1. The behaviour of the *Pipe Example* device as seen from the user via the EPICS interface is compliant to the NDS standard. Therefore, one can use this test plan also as the baseline for a test plan of an actual hardware device, but providing different means of verification of results.

This test plan assumes that if an operation succeeds against the Pipe Example dummy device, then the NDS that facilitates that function would also function correctly against a non-dummy device.

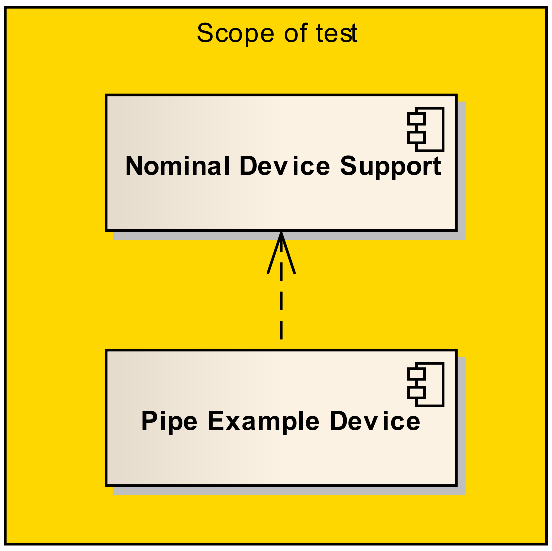

**Figure 1: Context diagram of the system under test.**

## 1.3 References

[RD1] Nominal Device Support Requirements Specification (NDS SRS) (9G4AZG v1.0)

[RD2] NDS source code: https://svnpub.iter.org/codac/iter/codac/contracts/2012/m-epics-nds

[RD3] IEEE Standard for Software and System Test Documentation (IEEE -829-2008)

[RD4] ITER CODAC Acronyms List (2LT73V v2.0)

[RD5] Nominal Device Support (NDS): EPICS Device Support Developer's Guide (66BSTM).

## 1.4 Definitions

For a complete list of ITER CODAC abbreviations see [RD4].

**Table 1-1: List of Abbreviations**

| | |
|---|---|
| **Acceptance testing** | Formal testing conducted to determine whether to accept a system or component. [RD3] |
| **Anomaly** | Anything observed in the documentation or operation of software or system that deviates from expectations based on previously verified software products, reference documents, or other sources of indicative behaviour. [RD3]<br>Synonyms: *deviation*, *bug*, *issue*, *problem*, *defect*, *fault*, *discrepancy*. |
| **CLI** | Command Line Interface |
| **CODAC** | Control, Data Access and Communication |
| **GUI** | Graphical User Interface |
| **HMI** | Human Machine Interface |
| **IDM** | ITER Document Management System |
| **IOC** | Input / Output Controller |
| **Integrity level** | The degree to which a system complies with a set of stakeholder-selected system characteristics (e.g., complexity, risk, safety level, security level, desired performance, reliability, or cost). [RD3] |
| **NDM** | Nominal Device Model |
| **NDS** | Nominal Device Support |
| **Test case** | Set of test inputs, execution conditions, and expected results developed for a particular objective. [RD3] |

| | |
|---|---|
| **Test item** | A software or system item that is an object of testing. [RD3] |
| **Test level** | A separate test effort that has its own documentation and resources (e.g., component, component integration, system, and acceptance). |
| **Test procedure** | Detailed instructions for the setup, execution, and evaluation of results for a given test case. [RD3] |
| **Test scenario** | A grouping of test cases. [RD3] |

# 2 DETAILS OF THE TESTING PROCESS

## 2.1 Definition of test levels

Following test levels are defined in this test plan to organize the testing activity.

| NDS Component Test | NDS |
|---|---|
| | |
| | |

## 2.2 Test administration

### 2.2.1 Test reporting requirements

Test reports shall be submitted in ITER Document Management System (IDM) using the Software Test Report Template [RD3] which includes test logs to describe the outcome of test procedures of the level test plans.

New errors or abnormal behaviour that are detected during the tests execution shall also be reported in the CODAC issue tracking system, Bugzilla [RD2] and referenced in the test report. This applies to errors in the tested products but not to errors in the tests that should only be reported in the test report.

In test report as well as in Bugzilla, a severity indication shall to be given (minor, normal, major, critical, blocker). The registered severity in Bugzilla will be reviewed with the bug by SCCB.

Tests defined in the test plan can refer to known bugs already registered in Bugzilla that have been assigned to be fixed at a given version of the tested software. In that case, the rest report shall include the outcome of a test with respect to these bugs. If the status of the bug was Resolved the outcome shall be Verified or Reopened. If a bug is not qualified as Resolved, its status cannot change not the tester may verify the behaviour with respect to the error description in the bug.

This test plan covers the following *Products/Components* as identified in Bugzilla:

- Product: Fast Controllers
- Component: Nominal Device Support

### 2.2.2 Test deliverables

Following documents shall be delivered in IDM:

- Software Test Report [NDS].

### 2.2.3 Estimates, cost and training

**Table 2-1: Skill level of testing team**

| Role/Qualifications | Time needed | Person |
|---|---|---|

| Role/Qualifications | Time needed | Person |
|---|---|---|
| **Quality Technician.**<br>■ Able to work with Linux terminal.<br>■ Familiar with EPICS concepts and tools.<br>■ Has access to ITER SVN. | 5 hours | |

# 3 NDS TEST PLAN

## 3.1 Scope

### 3.1.1 Test items and their identifiers

The Test Items being tested in the NDS Test Plan are

| Test Item | Major Version | Description |
|---|---|---|
| **NDS** | v.2.3.4 | |

### 3.1.2 Features to be tested

NDS library interface will be tested.

### 3.1.3 Features not to be tested

Tests are not cooperates with real data acquisition hardware or real time system. Only NDS interface is to be tested through the following test procedures.

### 3.1.4 Test traceability matrix

The system/software requirements being tested in Test Plan are

| Identifier | Requirement Title | Milestone | Test Procedure |
|---|---|---|---|
| NDS-SRS-F-001 | **Default settings** | 2.0 | TP-NDS-004 |
| NDS-SRS-F-002 | **Set device mode** | 2.0 | TP-NDS-002 |
| NDS-SRS-F-003 | **Set channel mode** | 2.0 | TP-NDS-003 |
| NDS-SRS-F-004 | **Read power status** | 2.0 | TP-NDS-002 |
| NDS-SRS-F-005 | **ERROR state** | 2.0 | |
| NDS-SRS-F-006 | **FAULT state** | 2.0 | |
| NDS-SRS-F-007 | **Turn device on** | 2.0 | TP-NDS-002 |
| NDS-SRS-F-008 | **Turn device off** | 2.0 | TP-NDS-002 |
| NDS-STS-F-009 | **Initialization** | 2.0 | TP-NDS-002 |
| NDS-STS-F-010 | **Initialization state** | 2.0 | TP-NDS-002 |

| | | | |
|---|---|---|---|
| NDS-SRS-F-011 | **Initialization status** | 2.0 | TP-NDS-002 |
| NDS-SRS-F-012 | **Load firmware** | 2.0 | TP-NDS-001 |
| NDS-SRS-F-013 | **Firmware version** | 2.0 | TP-NDS-001 |
| NDS-SRS-F-014 | **Firmware status** | 2.0 | TP-NDS-001 |
| NDS-SRS-F-015 | **Soft reset** | 2.0 | TP-NDS-005 |
| NDS-SRS-F-016 | **Master reset** | 2.0 | TP-NDS-005 |
| NDS-SRS-F-017 | **Power cycle** | 2.0 | TP-NDS-002 |
| NDS-SRS-F-020 | **Set clock source** | 2.0 | |
| NDS-SRS-F-021 | **Set clock frequency** | 2.0 | TP-NDS-DAQ-002 |
| NDS-SRS-F-022 | **Set clock multiplier** | 2.0 | |
| NDS-SRS-F-023 | **Temporal resolution** | 2.0 | |
| ~~NDS-SRS-F-024~~ | ~~**Trigger condition as absolute time**~~ | ~~2.0~~ | |
| ~~NDS-SRS-F-025~~ | ~~**Trigger condition as time delay**~~ | ~~2.0~~ | |
| ~~NDS-SRS-F-026~~ | ~~**Time trigger period**~~ | ~~2.0~~ | |
| NDS-SRS-F-027 | **Level triggering** | 2.0 | TP-NDS-DAQ-005 |
| NDS-SRS-F-028 | **Edge triggering** | 2.0 | |
| NDS-SRS-F-029 | **Configure trigger combinatorial logic** | 2.0 | TP-NDS-DAQ-005 |
| NDS-SRS-F-030 | **Trigger delay** | 2.0 | TP-NDS-DAQ-006 |
| NDS-SRS-F-031 | **Pre-trigger** | 2.0 | TP-NDS-DAQ-007 |
| NDS-SRS-F-032 | **Sampling duration – number** | 2.0 | |

| | | | |
|---|---|---|---|
| | **of samples** | | |
| NDS-SRS-F-033 | **Sampling duration - number of seconds** | 2.0 | |
| NDS-SRS-F-034 | **Continuous and one-shot trigger** | 2.0 | |
| NDS-SRS-F-035 | **Software triggering** | 2.0 | TP-NDS-DAQ-005 |
| NDS-SRS-F-040 | **Analog input filter** | TBD | not applicable |
| NDS-SRS-F-041 | **Image channel filter** | TBD | not applicable |
| NDS-SRS-F-042 | **Software filtering** | TBD | not applicable |
| NDS-SRS-F-043 | **Set size of the frame for FT** | TBD | not applicable |
| NDS-SRS-F-044 | **Set overlap** | TBD | not applicable |
| NDS-SRS-F-045 | **Set windowing function** | TBD | not applicable |
| NDS-SRS-F-046 | **Set smoothing factor** | TBD | not applicable |
| NDS-SRS-F-047 | **Software FT** | TBD | not applicable |
| NDS-SRS-F-048 | **Decimation on an input channel** | TBD | not applicable |
| NDS-SRS-F-049 | **Decimation on an output channel** | TBD | not applicable |
| NDS-SRS-F-050 | **Decimation offset** | TBD | not applicable |
| NDS-SRS-F-051 | **Conversion on an input channel** | TBD | not applicable |
| NDS-SRS-F-052 | **Conversion on an output channel** | TBD | not applicable |
| NDS-SRS-F-053 | **Default conversion** | TBD | not applicable |
| NDS-SRS-F-054 | **Function generator** | TBD | not applicable |

| | | | |
|---|---|---|---|
| NDS-SRS-F-055 | **Image compression** | TBD | not applicable |
| NDS-SRS-F-056 | **Quality flag** | 2.0 | |
| NDS-SRS-F-057 | **Quality level** | TBD | not applicable |
| NDS-SRS-F-060 | **Buffer size** | 2.0 | |
| NDS-SRS-F-070 | **Streaming parameters** | TBD | TP-NDS-TST-0009 |
| NDS-SRS-F-071 | **Streaming format** | TBD | TP-NDS-TST-0009 |
| NDS-SRS-F-072 | **Continuous streaming** | TBD | not applicable |
| NDS-SRS-F-073 | **Streaming on demand** | TBD | not applicable |
| NDS-SRS-F-080 | **Digital IO direction** | 2.0 | |
| NDS-SRS-F-081 | **Voltage levels** | 2.0 | |
| NDS-SRS-F-082 | **Digital channel states** | 2.0 | TP-NDS-003 |
| NDS-SRS-F-090 | **Set warnings and alarms** | 2.0 | |
| NDS-SRS-F-091 | **SEU detection** | 2.0 | |
| NDS-SRS-F-092 | **Cooling** | 2.0 | |
| NDS-SRS-F-093 | **Alarms and warnings status** | 2.0 | |
| NDS-SRS-F-094 | **Noise performance** | TBD | not applicable |
| NDS-SRS-F-095 | **Measurement quality** | 2.0 | |
| NDS-SRS-F-096 | **Correlation** | TBD | not applicable |
| NDS-SRS-F-097 | **Saturation** | TBD | not applicable |
| NDS-SRS-F-098 | **DAQ health log** | 2.0 | TP-NDS-INS-003 |
| NDS-SRS-F-099 | **Test** | 2.0 | TP-NDS-TST-0008 |

| | | | |
|---|---|---|---|
| NDS-SRS-F-100 | **Test type** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-F-101 | **Test verbose option** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-F-102 | **Test result identification** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-F-103 | **Test result description** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-F-104 | **Test result code** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-001 | **Send messages** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-002 | **Define message** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-003 | **Receive messages** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-004 | **Request-response** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-005 | **Predefined message types** | 2.0 | TP-NDS-TST-0008 |
| NDS-SRS-I-010 | **Device model** | 2.0 | TP-NDS-001 |
| NDS-SRS-I-011 | **Device serial number** | 2.0 | TP-NDS-001 |
| NDS-SRS-I-012 | **HW revision ID** | 2.0 | TP-NDS-001 |
| NDS-SRS-I-013 | **SW version** | 2.0 | TP-NDS-001 |
| NDS-SRS-I-014 | **Firmware version** | 2.0 | TP-NDS-001 |
| NDS-SRS-I-015 | **Device power** | 2.0 | |
| NDS-SRS-I-016 | **Power limit voltage** | 2.0 | |
| NDS-SRS-I-017 | **Device temperature** | 2.0 | |
| NDS-SRS-I-018 | **Device current** | 2.0 | |
| NDS-SRS-I-019 | **Device voltage** | 2.0 | |
| NDS-SRS-I-020 | **Set gain** | 2.0 | |

| | | | |
|---|---|---|---|
| NDS-SRS-I-021 | **Return gain** | 2.0 | |
| NDS-SRS-I-022 | **Set channel offset voltage** | 2.0 | |
| NDS-SRS-I-023 | **Set offset** | 2.0 | |
| NDS-SRS-I-024 | **Return the measured offset** | 2.0 | |
| NDS-SRS-I-025 | **Set bandwidth** | 2.0 | |
| NDS-SRS-I-026 | **Return bandwidth** | 2.0 | |
| NDS-SRS-I-027 | **Set Resolution** | 2.0 | |
| NDS-SRS-I-028 | **Set Image Type** | 2.0 | |
| NDS-SRS-I-029 | **Get samples per pixel** | 2.0 | |
| NDS-SRS-I-030 | **Set impedance** | 2.0 | |
| NDS-SRS-I-031 | **Return impedance** | 2.0 | |
| NDS-SRS-I-032 | **Set coupling** | 2.0 | |
| NDS-SRS-I-033 | **Return coupling** | 2.0 | |
| NDS-SRS-I-034 | **Set the type of input** | 2.0 | |
| NDS-SRS-I-035 | **Check the type of input** | 2.0 | |
| NDS-SRS-I-036 | **Ground input** | 2.0 | |
| NDS-SRS-I-037 | **Check if channel is grounded** | 2.0 | |

The registered bugs that are tested in this part are:

| Bug ID | Bug Description | Test Procedure |
|---|---|---|
| Bug 4583 | NDS: Ctrl-break handling failure | **TP-NDS-006** |
| Bug 4413 | NDS - [Mariusz] project generation and packaging | **TP-NDS-INS-003** |

| | | |
|---|---|---|
| Bug 4415 | NDS - [Mariusz] device initialization | **TP-NDS-002** |
| Bug 4418 | NDS - getFirmwareVersion | **TP-NDS-001** |
| Bug 4419 | NDS – packaging | **TP-NDS-INS-001** |
| Bug 4554 | NDS: OUT record's getter behavior on IOC initialization | **TP-NDS-004** |
| Bug 4584 | NDS: Steps in case of Device::createStructure() fails | |
| Bug 4615 | NDS: Implementing reset functionality [Mariusz Orlikowski] | **TP-NDS-002** |
| Bug 4618 | NDS - [Mariusz] Channel reset state requested. | **TP-NDS-002** |

## 3.2 Approach

### 3.2.1 Testing Methods

NDS is a user space library. A BlackBox approach is used for testing. Example application is using NDS library so testing is done through example application.

### 3.2.2 Item pass/fail criteria

Overall criteria for each test item to pass or fail:

| Test Item | Criteria |
|---|---|
| NDS | All requested function should be completed without an error. Debug output is allowed. |

### 3.2.3 Suspension criteria and resumption requirements

Testing shall be suspended immediately after more anomalies are discovered in any of the severity categories. Testing shall resume after all the anomalies are fixed as directed by the SCCB.

### 3.2.4 Task iteration policy

All test procedures of a Test Procedure Category shall be repeated where a Major or Medium anomaly was earlier reported. Testing of all other dependant groups or features shall be repeated.

### 3.2.5 Test coverage

The test coverage of the Test Procedures in this Test Plan is approximately

| | |
|---|---|
| Code coverage (manual tests) | **70%** |

| | |
|---|---|
| Functional coverage (manual tests) | **50%** |
| Requirements coverage | - |

## 3.3 Test Environment

### 3.3.1 Hardware

**Table 3-1: List of hardware in the test environment**

| ID | Hardware item | Location |
|---|---|---|
| **`Node`** | A host computer running CODAC. | |

### 3.3.2 Software

**Table 3-2: List of software in the test environment**

| Software component | Version | Where installed |
|---|---|---|
| CODAC | 4.2 | **`Node`** |
| NDS | 2.3.5 | **`Node`** |

The following tools are needed for testing:

- Shell script **`/opt/codac/tests/nds/ndsPipeExampleSignal.sh`.** The shell script generates:
  - AI0: a triangular signal ranging from 0 to 10, and back down to 0. Value changes by 0.1 at approximately 10Hz.
  - AI1: a triangular signal ranging from -100 to 100, and back down to -100. Value changes by 1 at approximately 10Hz.

These tools are installed as part of the **`codac-core-4.2-epics-nds-tests-4.1.0.v2.3`** package.

## 3.4 Test Procedures

### 3.4.1 Installation Tests

| **TP-NDS-INS-001** | **Verify Build** |
|---|---|
| **Description** | This tests compiles NDS source code, runs automated tests and prepares installable packages (RPMs). |
| **Pre-requisite** | |
| **1.** CODAC 4.1 machine in Mini-CODAC development role. | |

2. NDS is not yet installed. Check by executing:

```
$ rpm -qa | grep nds | grep codac
```

which should return an empty output.

| **Procedure** | |
|---|---|

1. Check-out NDS source code:

```
$ svn co https://svnpub.iter.org/codac/iter/codac/dev/units/m-epics-nds/ m-epics-nds
```

(Required version of NDS to test should be selected. For testing the development branch, `trunk` should be used. For testing the release version, choose the appropriate version in the `tags` folder.)

2. Build package

```
$ cd m-epics-nds
$ mvn package
```

| **Pass Criteria** | |
|---|---|

There are no errors during the build process. If there are warnings, report it as a comment below.

| **TP-NDS-INS-002** | **Install Procedure** |
|---|---|
| **Description** | This test confirms that NDS is installed. |
| **Pre-requisite** | TP-NDS-INS-001 |

1. NDS packages are uninstalled.

| **Procedure** | |
|---|---|

1. Check that the version of the NDS is properly reflected in the names of the RPM packages:

```
$ ls -l target/*.rpm
-rw-rw-r-- 1 codac-dev codac-dev 18991588 Mar 30 04:26 codac-core-4.2-
epics-nds-4.2.0.v2.3.3-1.el6.x86_64.rpm
…
```

**Expected result:** Second version number (after “v”) is 2.1. If this is not the case, report it in a comment below.

2. As **root**, install NDS

```
# rpm -ihv target/*.rpm
Preparing...                ########################################### [100%]
   1:codac-core-4.2-epics-nd########################################### [ 20%]
   2:codac-core-4.2-epics-nd########################################### [ 40%]
   3:codac-core-4.2-epics-nd########################################### [ 60%]
   4:codac-core-4.2-epics-nd###################################### [ 80%]
   5:codac-core-4.2-epics-nd###################################### [100%]
```

**3.** Check that packages are installed:

```
# rpm -qa | grep nds | grep codac
codac-core-4.2-epics-nds-devel-4.2.0.v2.3.3-1.el6.x86_64
codac-core-4.2-epics-nds-debuginfo-4.2.0.v2.3.3-1.el6.x86_64
codac-core-4.2-epics-nds-doc-4.2.0.v2.3.3-1.el6.x86_64
codac-core-4.2-epics-nds-src-4.2.0.v2.3.3-1.el6.x86_64
codac-core-4.2-epics-nds-4.2.0.v2.3.3-1.el6.x86_64
```

**Note:** Output of the commands could vary depending on testing NDS version and used CODAC version. Commands above shows package names for CODAC v.4.2.0 and NDS v.2.3.3

| | |
|---|---|
| **Pass Criteria** | |
| Results of checking installed packages are as indicated. | |
| | |
| **TP-NDS-INS-003** | **Create and run an example EPICS application** |
| **Requirement(s)** | NDS-SRS-F-098 |
| **Description** | In this test a skeleton driver, an example EPICS application and an IOC for the example application are created. |
| **Pre-requisite** | TP-NDS-INS-002 |
| **Procedure** | |

1. Change to an appropriate directory (e.g., the top directory for testing session) and create the unit:

```
$ mvn iter:newunit -Dunit=m-epics-pipeexample
[INFO] MODULE UNIT 'epics-pipeexample' CREATED
```

2. Create an example EPICS application:

```
$ cd m-epics-pipeexample
$ mvn iter:newapp -Dapp=pipeexample -Dtype=nds
[INFO] EPICS APPLICATION ‘pipeexample’ CREATED
```

3. Create an IOC that uses the application:

```
$ mvn iter:newioc -Dioc=pipeexample -Dapp=pipeexample -Dtype=nds
Using target architecture linux-x86_64 (only one available)
[INFO] EPICS IOC 'pipeexample' CREATED
[INFO] IOC 'pipeexample' INCLUDED FOR PACKAGING
```

4. Build the project:

```
$ mvn compile
```

5. Run the IOC:

```
$ mvn run
```

6. Check IOC console for debug output (NDS-SRS-F-098)

```
sevr=info [INF] ../pipeexampleDevice.cpp:421 getPower is called.
sevr=info [INF] ../pipeexampleDevice.cpp:433 setPower is called : 1
```

| | |
|---|---|
| **Pass Criteria** | |
| No errors | |

| **TP-NDS-INS-004** | **Uninstall Procedure** |
|---|---|
| **Description** | Check that uninstall of NDS is successful and cleans-up the system.<br>**Note**: This should be the last test that is executed, since tests that follow require NDS to be installed. |
| **Pre-requisite** | TP-NDS-INS-002 |
| **Procedure** | |

1. Uninstall NDS (as **root**)

```
# rpm -e --nodeps `rpm -qa | grep nds | grep codac`
```

| | |
|---|---|
| **Pass Criteria** | |
| There are no errors. | |

## 3.4.2 General NDS Functions Tests

| **TP-NDS-001** | **Firmware update** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-012 |
| **Description** | Test verifies firmware update functionality. NDS provides business logic to parse a meta-file, download the image if it is needed and call a user defined update procedure. |
| **Pre-requisite** | |
| Test case TP-NDS-INS-003 has passed. | |
| **Procedure** | |

1. Check hardware model:

```
$ caget PIPE-IMDL
PIPE-IMDL                        NI6529
```

2. Check firmware version:

```
$ caget PIPE-IFW
PIPE-IFW                       2.2.0
```

3. Check software version:

```
$ caget PIPE-ISW
PIPE-ISW                       N/A
```

4. Check hardware revision:

```
$ caget PIPE-IHW
PIPE-IHW                       rev0
```

**5.** Start update procedure:

```
$ caput -S PIPE-FWUP /opt/codac/tests/nds/data/chain.xml
```

**6.** Check firmware version:

```
$ caget PIPE-IFW
PIPE-IFW                       2.3.1
```

| **Pass Criteria** | |
|---|---|

1. There are no error messages in the IOC console.
2. "Updating firmware" message displayed in the IOC console:

```
sevr=info [INF] ../tstDevice.cpp:321 Updating firmware.
sevr=info [INF] ../tstDevice.cpp:323 Module to update: fpga-3 Current
firmware version: 2.2.0
sevr=info [INF] ../tstDevice.cpp:324 Image file:
/opt/codac/tests/nds/data/ni6529-fw-2.2.1.bin
…
```

| **TP-NDS-002** | **Device state machine testing** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-002 - **Set device mode**<br>NDS-SRS-F-007 - **Turn device on**<br>NDS-SRS-F-008 - **Turn device off** |
| **Bug(s)** | Bug 4615 |
| **Description** | Test verifies states of the device components and ability to switch states. |
| **Pre-requisite** | |

Test case TP-NDS-INS-003 has passed.

**Procedure**

1. Check device state:

```
$ camonitor PIPE
PIPE                           2014-03-03 12:57:38.350731 OFF
```

2. Switch ON device (use another console as far first one is blocked by camonitor option):

```
$ caput -S PIPE-MSGS ON
Old : PIPE-MSGS \003
New : PIPE-MSGS ON
```

Note: Old value could vary. It is not an error.

Expected monitoring output is:

```
PIPE                           2014-03-03 12:57:38.350731 OFF
PIPE                           2014-03-03 12:57:39.990408 INIT
PIPE                           2014-03-03 12:57:39.990416 ON
```

3. Check device state. Expected ON:

```
$ caget PIPE
PIPE                 ON
```

4. Switch OFF device:

```
$ caput -S PIPE-MSGS OFF
Old : PIPE-MSGS ON
New : PIPE-MSGS OFF
```

5. Switch ON device:

```
$ caput -S PIPE-MSGS ON
Old : PIPE-MSGS OFF
New : PIPE-MSGS ON
```

6. Reset device:

```
$ caput -S PIPE-MSGS RESET
```

Expected: Device state goes through RESETTING state.

```
PIPE                 2013-10-26 16:01:10.783598 ON
PIPE                 2013-10-26 16:01:23.407417 RESETTING
PIPE                 2013-10-26 16:01:28.407748 ON
```

**Pass Criteria**

1. There are no error messages in the IOC console.

| | |
|---|---|
| | |
| **TP-NDS-003** | **ChannelGroup state machine testing** |
| **Requirement(s)** | NDS-SRS-F-003 Set channel mode<br>NDS-SRS-F-082 Digital channel states |
| **Description** | Test verifies states of the device components and ability to switch states. |
| **Pre-requisite** | |

Test case TP-NDS-INS-003 has passed.

**Procedure**

1. Disable ChannelGroup

```
caput PIPE-AI-ENBL    0
Old : PIPE-AI-ENBL         ENABLED
New : PIPE-AI-ENBL         DISABLED
```

Note: Old value could vary. It is not an error.

2. Disable Channel

```
caput PIPE-AI0-ENBL 0
Old : PIPE-AI0-ENBL        ENABLED
New : PIPE-AI0-ENBL        DISABLED
```

Note: Old value could vary. It is not an error.

3. Switch device ON

```
$ caput -S PIPE-MSGS ON
Old : PIPE-MSGS /003
New : PIPE-MSGS ON
```

Note: Old value could vary. It is not an error.

**4.** Check device state. Expected ON:

```
$ caget PIPE
PIPE                 ON
```

**5.** Check ChannelGroup state. Expected DISABLED:

```
$ caget PIPE-AI-STAT
PIPE-AI-STAT         DISABLED
```

**6.** Check Channel state. Expected DISABLED:

```
$ caget PIPE-AI0-STAT
```

```
PIPE-AI0-STAT        DISABLED
```

**7.** Start processing on channel group:

```
$ caput -S PIPE-AI-MSGS START
Old : PIPE-AI-MSGS \003
New : PIPE-AI-MSGS START
```

**8.** Check ChannelGroup state. Expected PROCESSING:

```
$ caget -s PIPE-AI-STAT
PIPE-AI-STAT          PROCESSING
```

**9.** Check Channel state. Expected DISABLED:

```
$ caget -s PIPE-AI0-STAT
PIPE-AI0-STAT        DISABLED
```

**10.** Stop ChannelGroup processing:

```
$ caput -S PIPE-AI-MSGS STOP
Old : PIPE-AI-MSGS START
New : PIPE-AI-MSGS STOP
```

**11.** Check Channel state. Expected DISABLED:

```
$ caget PIPE-AI-STAT
PIPE-AI-STAT        DISABLED
```

**12.** Enable channel:

```
$ caput PIPE-AI0-ENBL 1
Old : PIPE-AI0-ENBL        DISABLED
New : PIPE-AI0-ENBL        ENABLED
```

**13.** Start processing on channel group:

```
$ caput -S PIPE-AI-MSGS START
```

**14.** Check ChannelGroup state PROCESSING:

```
$ caget PIPE-AI-STAT
PIPE-AI-STAT          PROCESSING
```

**15.** Check Channel state. Expected processing:

```
$ caget PIPE-AI0-STAT
PIPE-AI0-STAT        PROCESSING
```

| | |
|---|---|
| **Pass Criteria** | |

| | |
|---|---|
| **1.** There are no error messages in the IOC console. | |
| | |
| **TP-NDS-004** | **Automatic channels start** |
| **Requirement(s)** | NDS-SRS-F-003 Set channel mode |
| **Description** | Test verifies that enabled ChannelGroups/Channels starts automatically when device is started. |
| **Pre-requisite** | |
| Test case TP-NDS-INS-003 has passed. | |
| **Procedure** | |

1. Enable ChannelGroup

```
caput PIPE-AI-ENBL 1
Old : PIPE-AI-ENBL         DISABLED
New : PIPE-AI-ENBL         ENABLED
```

Note: Old value could vary. It is not an error.

2. Enable Channel

```
caput PIPE-AI0-ENBL 1
Old : PIPE-AI0-ENBL        DISABLED
New : PIPE-AI0-ENBL        ENABLED
```

Note: Old value could vary. It is not an error.

3. Switch device ON

```
$ caput -S PIPE-MSGS ON
Old : PIPE-MSGS /003
New : PIPE-MSGS ON
```

Note: Old value could vary. It is not an error.

**4.** Check device state. Expected ON:

```
$ caget PIPE
PIPE                 ON
```

**5.** Check ChannelGroup state. Expected PROCESSING:

```
$ caget PIPE-AI-STAT
PIPE-AI-STAT         PROCESSING
```

**6.** Check Channel state. Expected PROCESSING:

```
$ caget PIPE-AI0-STAT
PIPE-AI0-STAT        PROCESSING
```

| **Pass Criteria** | |
|---|---|

1. There are no error messages in the IOC console.

| **TP-NDS-005** | **Getter's behaviour on initialization** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-001 |
| **Bug(s)** | Bug 4554 |
| **Description** | The EPICS record getter is called over the initialization. NDS should allow the developer to handle the call. This test verifies that the getter is called and that data in the DB is not overwritten. |
| **Pre-requisite** | |

1. Test case **TP-NDS-INS-003** has passed.

| **Procedure** | |
|---|---|

Check IOC console output

| **Pass Criteria** | |
|---|---|

The IOC console should have following output:

```
[WARNING] sevr=info [INF] ../ndsPipeExampleDevice.cpp:397 getPower is
called.
[WARNING] sevr=info [INF] ../ndsPipeExampleDevice.cpp:409 setPower is
called : 1
```

The first line verifies that the getter was called during the initialization procedure. The second line confirms that the value in the database was not overwritten by the first call and stays the same as it was set in DB.

| **TP-NDS-006** | **Ctrl-break failure** |
|---|---|
| **Requirement(s)** | |
| **Bug(s)** | Bug 4583 |
| **Description** | This bug describes that during exit from an NDS application, a core dump is generated. This test verifies that Ctrl-C is handled correctly. |
| **Pre-requisite** | |

1. Test case **TP-NDS-INS-003** has passed.

**Procedure**

1. Switch ON device:

```
$ caput -S PIPE-MSGS ON
```

2. Switch ON channel:

```
$ caput -S PIPE-AO1-MSGS ON
```

Break execution by pressing Ctrl-C.

**Pass Criteria**

Application should be closed correctly without a core dump generation.

### 3.4.3 Data Acquisition Tests

| **TP-NDS-DAQ-001** | **Output a value** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | Test the functionality of outputting a value to the analog output channel. |

**Pre-requisite**

1. Test case **TP-NDS-004** has passed.
2. Switch on device:

```
$ caput -S PIPE-MSGS ON
```

3. Check device status

```
$ caget PIPE
PIPE                     ON
```

**Procedure**

1. In the **first** terminal set desired frequency (2Hz)

```
$ caput PIPE-AO0-CLKF 2
```

2. Start data acquisition on selected channel

```
$ caput -S PIPE-AO0-MSGS START
```

3. In the **second** terminal, monitor values on the channel analog output 0:

```
$ cat /tmp/q.ao.0
```

**Note:** File created only after the channel start processing.

4. Set the value of the record corresponding to the channel analog output 0:

```
$ caput PIPE-AO0-OUT 17.23
```

| **Pass Criteria** | |
|---|---|

Value 17.23 should appear in the first terminal.

| **TP-NDS-DAQ-002** | **Playing-out a waveform** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-021 |
| **Description** | Test the functionality of outputting values from a waveform record to the corresponding analog output channel. |
| **Pre-requisite** | |

1. Test case **TP-NDS-INS-003** has passed.
2. Switch on device:

```
$ caput -S PIPE-MSGS ON
```

Check device status

```
$ caget PIPE
PIPE                     ON
```

| **Procedure** | |
|---|---|

1. In the **second** terminal set the values of the waveform record (e.g. let values range from 1 to 200). Default size of a channel's waveform record is 200.

```
$ caput -a PIPE-AO0 1 `seq 1 200`
```

2. In the second terminal, set the frequency of the internal clock (e.g. 10.5 Hz) at which the values from the waveform record are to be written on the pipe.

```
$ caput PIPE-AO0-CLKF 10.5
```

3. Start data processing on selected channel.

```
$ caput -S PIPE-AO0-MSGS START
```

**4.** In the **first** terminal monitor the values on the output channel:

```
$ cat /tmp/q.ao.0
```

Note: File created only after the channel start processing.

**Expected result:**

1. In the first terminal, approximately 10 new consecutive integer values should appear every second.
2. Maximum value should be 200 and minimum value should be 1.
3. Value 200 is immediately followed by number 1, after which the sequence repeats.
4. Stop generation of the waveform:

```
$ caput -S PIPE-AO0-MSGS STOP
```

**Expected result:** The first terminal stops producing values.

**Pass Criteria**

1. In the first terminal, approximately 10 new consecutive integer values should appear every second (10Hz).
2. Maximum value should be 200 and minimum value should be 1.
3. Value 200 is immediately followed by number 1, after which the sequence repeats.

| **TP-NDS-DAQ-003** | **Reading a value** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | Test the functionality of reading individual values from analog input channels. |
| **Pre-requisite** | |

1. Test case TP-NDS-001 has passed.
2. Switch on device:

```
$ caput -S PIPE-MSGS ON
```

3. Check device status

```
$ caget PIPE
PIPE                     ON
```

| | |
|---|---|
| **Procedure** | |

1. Enable data acquisition for selected terminal:

```
$ caput -S PIPE-AI0-MSGS START
```

In the **first** terminal monitor the record with suffix IN that corresponds to channel analog input 0:

```
$ camonitor PIPE-AI0-IN
```

In the **second** terminal, write a value to the channel:

```
$ echo 88.33 > /tmp/q.ai.0
```

Note: File created only after the channel start processing.

| | |
|---|---|
| **Pass Criteria** | |

In the first terminal, the entered value appears.

| | |
|---|---|
| **TP-NDS-DAQ-004** | **Waveform sampling on channel** |
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | Test the functionality of sampling values from an analog input into a waveform record. |
| **Pre-requisite** | |

1. Test case **TP-NDS-INS-003** has passed.
2. Switch on device:

```
$ caput -S PIPE-MSGS ON
```

3. Check device status

```
$ caget PIPE
PIPE                     ON
```

| | |
|---|---|
| **Procedure** | |

1. In the **first** terminal set the frequency of internal clock for channel analog input 0. Size of the dummy device buffer is by default 200 elements, therefore setting the sampling frequency at 100 Hz, causes the buffer to be filled every 2 seconds.

```
$ caput PIPE-AI0-CLKF 100
```

2. Start data acquisition for selected terminal:

```
$ caput -S PIPE-AI0-MSGS START
```

3. In the **first** terminal monitor the waveform record of analog input channel 0 (size of the waveform record is by default 200 elements):

```
$ camonitor PIPE-AI0
```

   Since at this point there should be no input on the channel, **`camonitor`** periodically (approximately every two seconds) returns a waveform full of constant values.

4. In the **second** terminal run the **`ndsPipeExampleSignal.sh`** auxiliary test script. For analog input 0, this script generates a triangle signal with maximum value 10, minimum value 0.

```
$ sh /opt/codac/tests/nds/ndsPipeExampleSignal.sh
```

**DO NOT CLEANUP IF CONTINUING WITH TRIGGERING TESTS BELOW. Rather, perform these steps after the last triggering test.**

1. Stop **`ndsPipeExampleSignal.sh`** script by pressing Ctrl+C in the **second** terminal.
2. Stop **`camonitor`** by pressing Ctrl+C in the **first** terminal.
3. Stop sampling on the channel:

```
$ caput -S PIPE-AI0-MSGS STOP
```

| **Pass Criteria** | |
|---|---|

In the **first** terminal, a waveform with consecutive values ranging from 0 to 10 should be updated every 2 seconds.

| **TP-NDS-DAQ-005** | **Level trigger on multi-channel condition** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-027<br>NDS-SRS-F-029<br>NDS-SRS-F-035 |

| | |
|---|---|
| **Bugs(s)** | [Optional][List bugs (hyperlinks) verified with this test procedure] |
| **Description** | Test level triggering on an analog input channel. Waveforms are generated on the first and second input channel, and level trigger is set when conditions on both channels are met. |
| **Pre-requisite** | |

1. Test case TP-NDS-DAQ-004 has passed and is left running (i.e., cleanup not performed).

**Procedure**

1. Establish the same setup as in TP-NDS-DAQ-004, but do not start data acquisition beforehand.
2. In a **third** terminal, set a trigger condition for channel analog input 0:

```
$ caput -S PIPE-AI0-TRGC "ai0>5 ai1<0"
```

3. Start data acquisition for selected terminal:

```
$ caput -S PIPE-AI0-MSGS START
$ caput -S PIPE-AI1-MSGS START
```

**Pass Criteria**

In the **`camonitor`** output, values of the waveform record should start updating every 2 seconds (time needed to fill the buffer) after the trigger condition is met – i.e., AI0 is above 5, and AI1 is negative. Refer to the output of **`ndsPipeExampleSignal.sh`**. After the trigger condition is no longer met, the updating should stop after the buffer is filled.

| | |
|---|---|
| **TP-NDS-DAQ-006** | **Trigger with a positive delay** |
| **Requirement(s)** | NDS-SRS-F-030 |
| **Description** | Test triggering with a time delay – i.e., start data acquisition at some time after the trigger condition is met. In this test, we test for a positive value of delay. |
| **Pre-requisite** | |

1. Test case TP-NDS-DAQ-0005 has passed.

**Procedure**

1. Establish the same setup as in TP-NDS-DAQ-004, but do not start data acquisition beforehand.
2. In the **third** terminal set the trigger condition for channel analog input 0:

```
$ caput -S PIPE-AI0-TRGC "AI0>5"
```

**3.** In the **third** terminal set the value of a trigger delay (number of seconds) on analog input 0:

```
$ caput PIPE-AI0-TRGD 4
```

**4.** Start data acquisition for selected terminal:

```
$ caput -S PIPE-AI0-MSGS START
$ caput -S PIPE-AI1-MSGS START
```

**Pass Criteria**

In the **camonitor** output, updating of the waveform record should start 6 seconds (delay + number of seconds needed to fill the buffer) after the trigger condition transitions from false to true. First sampled value should be 8.6 ± 0.1.

After the trigger condition is no longer met, the updating should stop. In the last waveform update, the values at the beginning of the buffer should still be above 5, and value at the end of the buffer is expected to be below 5 (around 4.4 ± 0.1, but depending on the exact starting condition).

| **TP-NDS-DAQ-007** | **Pre-trigger** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-031 |
| **Bugs(s)** | [Optional][List bugs (hyperlinks) verified with this test procedure] |
| **Description** | Test the functionality of pre-triggering on an analog input channel. Pre-triggering means that samples before the trigger condition was met are also captured. This corresponds to a negative value of the triggering delay. |

**Pre-requisite**

**1.** Test case TP-NDS-DAQ-0005 has passed.

**Procedure**

**1.** Repeat the steps of test TP-NDS-DAQ-0004.

**2.** In the **third** terminal set the trigger condition for channel analog input 0:

```
$ caput -S PIPE-AI0-TRGC "AI0>5"
```

**3.** In the **third** terminal set the value of a trigger delay (number of seconds) on analog input 0:

```
$ caput PIPE-AI0-TRGD -4
```

| | |
|---|---|
| **4.** Run the cleanup steps described of the test case TP-NDS-DAQ-004. | |
| **Pass Criteria** | |
| In the **`camonitor`** output, updating of the waveform record should start 2 seconds after the trigger condition is met, however sampled values should be 4 seconds behind the current value on the channel (first sampled value should be 1.5 ± 0.1).<br>After the trigger condition is no longer met, the sampling should stop. Thus, the last sampled buffer will have initial values greater than 5, and last values below 5 (last sampled value should be 4.2 ± 0.1). | |

### 3.4.4 Timing Features Tests

| TP-NDS-TST-0001 | Create and run a timing EPICS application |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Bugs(s)** | [Optional][List bugs (hyperlinks) verified with this test procedure] |
| **Description** | In this test a skeleton driver of a timing device, an example EPICS application and an IOC for the example application are created. |
| **Pre-requisite** | |
| **1.** NDS is installed [TP-NDS-INS-0002] | |
| **Procedure** | |

**1.** Create module unit

```
$ mvn iter:newunit -Dunit=m-epics-timing
[INFO] MODULE UNIT 'm-epics-timing' CREATED
```

**2.** Create an example EPICS application

```
$ cd m-epics-timing
$ mvn iter:newapp -Dapp=timing -Dtype=ndsTime
[INFO] EPICS APPLICATION 'timing' CREATED
```

**3.** Create IOC

```
$ mvn iter:newioc -Dioc=timing -Dapp=timing -Dtype=ndsTime
Using target architecture linux-x86_64 (only one available)
[INFO] EPICS IOC 'timing' CREATED
[INFO] IOC 'timing' INCLUDED FOR PACKAGING
```

**4.** Build application

```
$ mvn compile
```

**5.** Run the IOC

```
$ mvn run
```

**6.** Start device processing

```
$ caput -S TM-MSGS ON
```

**7.** Check device status

```
$ caget TM
TM                  ON
```

| **Pass Criteria** | |
|---|---|
| There are no errors. | |
| **TP-NDS-TST-0002** | **Check timestamping on demand** |
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | This test validates the ability of NDS to set the time stamp of a record on demand. A record's update time is set from a read handler. |
| **Pre-requisite** | |
| **1.** Sample application is run [TP-NDS-TST-0001] | |
| **Procedure** | |

**1.** Run time monitoring command

```
$camonitor TM-BDTM
```

**2.** Force record to be updated

```
$caput TM-BDTM.PROC 1
```

**3.** Check that record obtain new timestamp

```
TM-TIME 2013-04-24 10:27:08.057811 2 7.3564e+08 5.78107e+07
TM-TIME 2013-04-24 10:27:09.057825 2 7.3564e+08 5.78248e+07
TM-TIME 2013-04-24 10:27:10.057880 2 7.3564e+08 5.78802e+07
TM-TIME 2013-04-24 10:27:11.057934 2 7.3564e+08 5.79335e+07
TM-TIME 2013-04-24 10:27:12.057988 2 7.3564e+08 5.79878e+07
…
```

| **Pass Criteria** | |
|---|---|
| There are no errors and the current time is returned. | |

| TP-NDS-TST-0003 | Check timestamp through interrupt |
|---|---|
| Requirement(s) | [Requirement Identifier(s) or brief description] |
| Description | This test checks the ability of NDS to set a time stamp through an EPICS interrupt handler and checks that the EPICS Timestamp is set when the interrupt handler is called to update a record's data. |
| Pre-requisite | |

1. Sample application is run [TP-NDS-TST-0001]

**Procedure**

1. Monitor for event updates

```
$ camonitor TM-PFI0-EVT  TM-PFI0-TS
TM-PFI0-TS 1990-01-01 22:30:00.000000 2 77400 0
TM-PFI0-EVT             1990-01-01 22:30:00.000000 129
```

2. Process timestamp

```
$ caput TM-BDTM.PROC 1
```

The TSEL field of the EVT record is linked to the TIME field of the TMST record, so the EVT record should be time stamped with the same time.

**Pass Criteria**

There are no errors and the timestamp of the TMST and EVT records are updated consecutively.

| TP-NDS-TST-0004 | Check handling of record's reason arguments |
|---|---|
| Requirement(s) | [Requirement Identifier(s) or brief description] |
| Description | NDS provides a way to pass arguments through the **`asyn`** reason, so additional parameters could be transferred to handlers. This test checks the ability to handle records of the same reason but with different arguments by the same handler. |
| Pre-requisite | |

1. Sample application is run [TP-NDS-TST-0001]

**Procedure**

1. Update record value with first reason's argument.

```
$ caput TM-PFI0-RSN1 1
```

Make sure that the first argument ('arg1') is printed in the IOC console.

```
epics> Arg[0]: arg1
```

2. Update the record value of the same reason but with the second reason's argument.

```
$ caput TM-PFI0-RSN2 1
```

Make sure that the first argument ('arg2') is printed in the IOC console.

```
epics> Arg[0]: arg2
```

| | |
|---|---|
| **Pass Criteria** | |
| There are no errors and both records are handled by the same setter. | |
| | |
| **TP-NDS-TST-0005** | **Check handling of FTE event parameters** |
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | NDS implements a generalization of the FTE event. This test verifies that the FTE event's (record's) handlers were called on update. |
| **Pre-requisite** | |
| 1. Sample application is run [TP-NDS-TST-0001] | |
| **Procedure** | |

1. Update record value with first reason's argument.

```
caput TM-FT1-D 2
caput TM-FT1-E 1
caput TM-FT1-L 0
```

Make sure that there is debug information in the IOC console:

```
epics>
FTE: setDelay EventID: 1 TerminalID: PXITRG1
FTE: setEnabled EventID: 1 TerminalID: PXITRG1
FTE: setLevel EventID: 1 TerminalID: PXITRG1
```

| | |
|---|---|
| **Pass Criteria** | |
| There are no errors and all records are handled correctly. | |
| | |

| **TP-NDS-TST-0006** | **Check handling of PULSE event parameters** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | NDS implements abstraction of the PULSE event. This test verifies that all record's handlers are called from the C++ level. |
| **Pre-requisite** | |

1. Sample application is run [TP-NDS-TST-0001]

**Procedure**

1. Update record value with first reason's argument.

```
caput TM-PL2-D 1
caput TM-PL2-W 3
caput TM-PL2-E 1
```

Make sure there is debug information in the IOC console:

```
epics>
Pulse: setDelay EventID: 13 TerminalID: PXISTAR2
Pulse: setWidth EventID: 13 TerminalID: PXISTAR2
Pulse: setEnabled EventID: 13 TerminalID: PXISTAR2
```

**Pass Criteria**

There are no errors and all records are handled correctly.

| **TP-NDS-TST-0007** | **Check handling of CLOCK event parameters** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | NDS implements an abstraction of the CLOCK event. NDS does not provide a CLOCK generator itself, so the test verifies that handlers for all records are called. |
| **Pre-requisite** | |

1. Sample application is run [TP-NDS-TST-0001].

**Procedure**

1. Update record value with first reason's argument.

```
caput TM-CK3-D   1
caput TM-CK3-DC  2
caput TM-CK3-P   4
caput TM-CK3-E   1
caput TM-CK3-TE  6
```

Make sure there is debug information in IOC console:

```
epics>
Clock: setDelay EventID: 21 TerminalID: PFI0
Clock: setDutyCycle EventID: 21 TerminalID: PFI0
Clock: setWidth EventID: 21 TerminalID: PFI0
Clock: setEnabled EventID: 21 TerminalID: PFI0
Clock: setTimeEnd EventID: 21 TerminalID: PFI0
```

**Pass Criteria**

There are no errors and all records are handled correctly.

| **TP-NDS-TST-0008** | **Check handling of TEST command for timing device** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-099<br>NDS-SRS-F-100<br>NDS-SRS-F-101<br>NDS-SRS-F-102<br>NDS-SRS-F-103<br>NDS-SRS-F-104<br>NDS-SRS-I-005 |
| **Description** | NDS provides stubs for the TEST message type. This test verifies that the TEST message handler was called when the TEST message is sent. |
| **Pre-requisite** | |

1. Sample application is run [TP-NDS-TST-0001].

**Procedure**

1. Run monitoring for response messages

```
$ camonitor -S TM-MSGR TM-PFI0-MSGR
```

2. Send TEST command to device instance

```
$ caput -S TM-MSGS TEST
Old : TM-MSGS \003
New : TM-MSGS TEST
```

Response to this command should be the following:

```
TM-MSGR 2013-04-25 15:31:19.751902 TEST CODE=-1,TEXT=Not implemented
```

**3.** Send TEST command to terminal instance

```
$ caput -S TM-PFI0-MSGS TEST
Old : TM-PFI0-MSGS \003
New : TM-PFI0-MSGS TEST
```

Response to this command should be the following:

```
TM-PFI0-MSGR 2013-04-25 15:31:37.738198 TEST CODE=0,TEXT=Success.
```

| **Pass Criteria** | |
|---|---|
| There are no errors and all records are handled correctly. | |

| **TP-NDS-TST-0009** | **Check handling of STRM record** |
|---|---|
| **Requirement(s)** | NDS-SRS-F-070<br>NDS-SRS-F-071 |
| **Description** | NDS provides stubs to configure streaming URL. NDS does not supports streaming, so this test only verifies the handling of this record. |
| **Pre-requisite** | |

**1.** Sample application is run [TP-NDS-TST-0001].

**Procedure**

**1.** Update record value with URL

```
caput -S TM-PFI0-STRM  https://www.google.com/
```

Make sure that debug information is displayed in IOC console:

```
[WARNING] sevr=minor [WRN] ../ndsTimingTerminal.cpp:127 Streaming URL is
set: https://www.google.com/
```

**Pass Criteria**

There are no errors and all records are handled correctly.

### 3.4.5 DAQ System testing

| TP-DAQ-SYS-0001 | Creating DAQ system application |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | Test verifies ability to combine DAQ and Timing device support to DAQ System. Test requires separate DAQ and Timing device support compiled and installed in the system. Installation must be performed in standard CODAC way. |
| **Pre-requisite** | |

1. Dummy DAQ device support application is instantiated [**TP-NDS-INS-003**].
2. Dummy Timing device support application is instantiated [TP-NDS-TST-0001].

**Procedure**

1. Preparing DAQ device support application for packaging. Add the following code to the pom.xml file of dummy DAQ device support application (see pre-requisite):

```
       <package name="lib">
           <include type="file"
            source="main/epics/lib"
            target="epics/modules/pipeexample/lib" />

           <include type="file"
            source="main/epics/db"
            target="epics/modules/pipeexample/db" />

           <include type="file"
            source="main/epics/dbd"
            target="epics/modules/pipeexample/dbd" />

       </package>
```

**Note:** this code must be added between the following tags:

```
<configuration>
   <packaging>
…
   </packaging>
</configuration>
```

2. Package DAQ device support library

```
$ mvn package
…
[INFO] Successfully packaged: codac-core-4.2-epics-pipeexample-lib-
4.2b4.v0.0a1-1.el6.x86_64.rpm
[INFO] PACKAGING COMPLETED (see for .rpm files under 'target' directory)
```

3. Install pipeexample-lib rpm. (**Note:** you must be a root)

```
# rpm -ihv target/codac-core-4.2-epics-pipeexample-lib-4.2b4.v0.0a1-
1.el6.x86_64.rpm
Preparing...                ###########################################
[100%]
   1:codac-core-4.2-epics-pi###########################################
[100%]
```

**4.** Preparing Timing device support application for packaging. Add the following code to the pom.xml file of timing device support application (see pre-requsite):

```
<package name="lib">
   <include type="file"
      source="main/epics/lib"
      target="epics/modules/timing/lib" />

   <include type="file"
      source="main/epics/db"
      target="epics/modules/timing/db" />

   <include type="file"
      source="main/epics/dbd"
      target="epics/modules/timing/dbd" />
</package>
```

**Note:** this code must be added between the following tags:

```
<configuration>
   <packaging>
…
   </packaging>
</configuration>
```

**5.** Package Timing device support library

```
$mvn package
[INFO] Successfully packaged: codac-core-4.2-epics-timing-lib-
4.2b4.v0.0a1-1.el6.x86_64.rpm
[INFO] PACKAGING COMPLETED (see for .rpm files under 'target' directory)
```

**6.** Install timing-lib rpm. (**Note:** you must be a root)

```
$rpm -ihv target/codac-core-4.2-epics-timing-lib-4.2b4.v0.0a1-
1.el6.x86_64.rpm
Preparing...                ###########################################
[100%]
   1:codac-core-4.2-epics-ti###########################################
[100%]
```

Now everything is ready to build DAQ System.

**7.** Instantiate DAQ System application

```
mvn iter:newunit -Dunit=m-epics-daqsys
cd m-epics-daqsys/
mvn iter:newapp -Dapp=daqsys
mvn iter:newioc -Dioc=daqsys -Dapp=daqsys -Dtype=nds
```

8. Edit **st.cmd**

```
vi src/main/epics/iocBoot/iocdaqsys/st.cmd
```

9. Add DAQ device support to Makefile.

```
ndsCreateDevice "pipeexample", "ndsTestPIPE",
"FILE=/tmp/q,N_AI=2,N_AO=3,N_DI=4,N_DO=5,N_DIO=6,N_IMAGE=7"

dbLoadRecords "db/pipeexample.db", "PREFIX=PIPE, IDX=0, MODULEIDX=0,
ASYN_PORT=ndsTestPIPE, TIMEOUT=1"
```

10. Add timing device support initialization:

```
ndsCreateDevice "timing", "ndsTimmvn runing", "TRG=2"
## Load record instances
dbLoadRecords "db/timing.db", "PREFIX=TM, IDX=0, MODULEIDX=0,
ASYN_PORT=ndsTiming, TIMEOUT=1"
```

11. Compile and run DAQ System

```
$mvn compile
$mvn run
```

There must be no error in EPICS console. DB files for both devices must be loaded successfully.

| **TP-DAQ-SYS-0002** | **Reading timestamp from timing terminal** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | NDS provides a way to configure timestamp source so timestamp for specific channel could be obtained from this source. Test verifies that timestamp could be obtained from timing terminal. |
| **Pre-requisite** | |

1. Sample application is run [TP-DAQ-SYS-0001].

**Procedure**

1. List registered devices. Execute command in IOC shell.

```
epics> ndsListDevices
ndsTestPIPE
ndsTiming
```

Make sure that required device loaded.

2. Configure timing device to obtain timestamps

```
caput PIPE-AI0-TDEV ndsTiming
```

3. Check if everything is OK:

```
$ caget PIPE-AI0-TDEV.STAT
PIPE-AI0-TDEV.STAT              NO_ALARM
```

4. Configure timestamping terminal to obtain timestamps

```
caput PIPE-AI0-TTRM pfi1
```

5. Force to update timestamp

```
caput PIPE-AI0-TS.PROC 1
```

6. Read timestamp from TS pv

```
$ camonitor PIPE-AI0-TS
PIPE-AI0-TS 1990-01-01 01:10:00.000000 2 600 0
```

7. Dummy Timing terminal provide timestamp with 10 minutes basis starting from EPICS epoch (1990-01-01 01:00:00.000000).

| **TP-DAQ-SYS-0003** | **Timestamping data** |
|---|---|
| **Requirement(s)** | [Requirement Identifier(s) or brief description] |
| **Description** | NDS provides a way to configure timestamp source so timestamp for specific channel could be obtained from this source. Test verifies that acquired data is timestamped with a timestamp from timing terminal. |
| **Pre-requisite** | |

1. Sample application is run [TP-DAQ-SYS-0001].
2. Data acquisition channel is configured to obtain timestamps from timing terminal [TP-DAQ-SYS-0002]

**Procedure**

1. Start DAQ device

```
$ caput -S PIPE-MSGS ON
```

2. In the **first** terminal set the frequency of internal clock for channel analog input 0. Size of the dummy device buffer is by default 200 elements, therefore setting the sampling

frequency at 100 Hz, causes the buffer to be filled every 2 seconds.

```
$ caput PIPE-AI0-CLKF 100
```

**3.** Start data acquisition for selected terminal:

```
$ caput –S PIPE-AI0-MSGS START
```

**4.** In the **first** terminal monitor the waveform record of analog input channel 0 (size of the waveform record is by default 200 elements):

```
$ camonitor PIPE-AI0
```

Since at this point there should be no input on the channel, **camonitor** periodically (approximately every two seconds) returns a waveform full of constant values.

**5.** In the **second** terminal run the **ndsPipeExampleSignal.sh** auxiliary test script. For analog input 0, this script generates a triangle signal with maximum value 10, minimum value 0.

```
$ sh /opt/codac/tests/nds/ndsPipeExampleSignal.sh
```

**6.** Check that timestamps for each buffer follows with 10 min gap. Dummy timing terminal provides fake timestamps with 10 minutes period starting from EPICS epoch.

```
PIPE-AI0 1990-01-01 02:50:00.000000 200 0.2 0.2 0.2 0.3 0.3 0.3 0.3 0.3
0.3 0.3 0.3 0.3 0.3 0.3 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.5
…
PIPE-AI0 1990-01-01 03:00:00.000000 200 2.1 2.1 2.1 2.2 2.2 2.2 2.2 2.2
2.2 2.2 2.2 2.2 2.2 2.2 2.3 2.3 2.3 2.3 2.3 2.3 2.3 2.3 2.3 2.3 2.3 2.4
…
```

**Note:**

1. Some values are cut from the output.
2. Timestamp of  2 consecutive buffers differ in 10 minutes.

## 3.5 NDS Test Log

### 3.5.1 Installation Tests

| | | |
|---|---|---|
| **TP-NDS-INS-001** | **Verify Build** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-INS-002** | **Install Procedure** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |

| | | |
|---|---|---|
| | | |
| **TP-NDS-INS-003** | **Create and run an example EPICS application** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-INS-004** | **Uninstall Procedure** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |

### 3.5.2 General NDS Functions Tests

| | | |
|---|---|---|
| **TP-NDS-001** | **Firmware update** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-002** | **Device state machine testing** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-003** | **ChannelGroup state machine testing** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-004** | **Automatic channels start** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-005** | **Getter's behaviour on initialization** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-006** | **Device reset** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |

| TP-NDS-007 | Ctrl-break failure | [PASS / FAIL] |
|---|---|---|
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |

### 3.5.3 Data Acquisition Tests

| TP-NDS-DAQ-001 | Output a value | [PASS / FAIL] |
|---|---|---|
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-002** | **Playing-out a waveform** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-003** | **Reading a value** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-004** | **Waveform sampling on channel** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-005** | **Level trigger on multi-channel condition** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-006** | **Trigger with a positive delay** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-DAQ-007** | **Pre-trigger** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |

### 3.5.4 Timing Features Tests

| TP-NDS-TST-0001 | Create and run a timing EPICS application | [PASS / FAIL] |
|---|---|---|
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |

| | | |
|---|---|---|
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0002** | **Check timestamping on demand** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0003** | **Check timestamp through interrupt** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0004** | **Check handling of reason's arguments** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0005** | **Check handling of FTE event parameters** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0006** | **Check handling of PULSE event parameters** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0007** | **Check handling of CLOCK event parameters** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0008** | **Check handling of TEST command for timing device** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |
| **TP-NDS-TST-0009** | **Check handling of STRM record** | [PASS / FAIL] |
| [Bug Identifier(s)] | [Bug title(s)] | [Severity] |
| [Anomaly report] | | |
| | | |