Space Details

Key:	AST
Name:	900626 Asset Configuration and Tracking Consolidation
Description:	
Creator (Creation Date):	kgomes (Oct 18, 2006)
Last Modifier (Mod. Date):	kgomes (Oct 18, 2006)

Available Pages

- Asset Configuration and Tracking Consolidation
  - Approaches
  - BogInstruNormalization
  - Design
    - AccessAppChanges
    - DatabaseFields
      - MakingData
        - SqlQueries
          - SqlQueriesFinal
  - Design Review Notes
  - Requirements
  - Use Cases

Asset Configuration and Tracking Consolidation

This page last changed on Oct 24, 2007 by kgomes.

Welcome to the project page for the Asset Configuration and Tracking Consolidation project.

Here are some related pages:

1. Use Cases
2. Requirements
3. Approaches
4. Modeling
5. Design

---

1. Design Review Notes
2. BogInstruNormalization
3. DatabaseFields
4. MakingData – detailed steps needed to transform the contents of the database

Tasks and bugs can be found in JIRA

Approaches

This page last changed on Nov 06, 2006 by graybeal.

There are three overarching approaches to this task:

1) **Merge** Make the necessary changes to merge the two databases (probably into SSDS) and make the merged data visible through the same interfaces (MS Access, SSDS) that are available now.

2) **Synchronize** Synchronize the two databases so that changes in each appear in the other (though not all fields have to appear in both). Perform synchronizations via triggers or DTS jobs as appropriate.
2a) **Dual create** Support new instrument entry on either database, synchronizing the entries as needed and validating on sync.
2b) **Unicreate** Support new instrument entry only on one database.

3) **Sync-to-SSDS** Push information from the Instru database on BOG to the SSDS database. Don't bother to update the BOG database with changes that have been made to the SSDS data. (This is like Paul "keeping his own copy," which is what we are doing today, but it provides SSDS with the information Paul enters.)

Option	Name	Pros	Cons	Comments
1	Merge	fairly maintainable minimizes duplicate info	lots of merging required requires significant control security causes most change	must track "devices of interest" to OSG
2	Synchronize	simplifies interface work	lots of model changes triggers fear of changes most state complexity	checkpoints needed for some synchronized data
2a	Sync-dual create	most like current system for users	lots more checkpointing	
2b	Sync-unicreate	simple to implement	reduces functionality to other interface	
3	Sync-to-SSDS	simplest number of changes least OSG impact	no 2-way interoperability	could report differences

BogInstruNormalization

This page last changed on Feb 27, 2007 by graybeal.

Introduction

This page describes normalization planned for various fields. The actual steps required to perform this normalization are documented in MakingData.

BOG fields for Model and Serial Numbers

Guidelines:

- Model: The available labels are not as well codified as Type, Manufacturer. As long as the two tables agree (initially), it doesn't matter if it's "37IM" or "37 IM" or "37-IM" etc; it's been suggested that tools for searching on model be case-insensitive, also ignoring spaces, hyphens. There's sure to be a lot of hand-alignment.
- The serial number should be the most complete serial number as presented and formatted by the vendor — as written on the invoice, written on the label, etc. This will often include some model information. (The short serial number in the Instru 'Serial' field is the last unique part of the full serial number. We will support that also but it will not be primary.)
- Name may be the Type or Model or a more descriptive name, potentially in alignment with that provided by the manufacturer. This information can be constructed from the other fields in any case, and so it is believed not critical to processing.

Some example formats for the different manufacturers are in this table.

Mannufacturer	Model	Serial Number	Name
Aanderaa Data Instruments	just number (3830, 3835)	just number (131, 121)	Name as provided by Aandreaa ("Oxygen Optode", "Oxygen Optode Shallow Water")
WHOI (was ASIMET)	three-letter abbrev from ASIMET (LWR, SWR)	three-digit number	"ASIMET " + model (e.g. "ASIMET SWR")
Benthos	as provided by company (865-A, UAT-376)	as provided by company	Description "release" or "transducer"
Biospherical	as provided by company (PRR-620, PRR-620-T2)	as provided by company	

To create the normalized SSDS serial number field, we plan the following.

BOG Field	Current Values	Suggested Value
FullSerial	A real mess	If Instru has a full serial number, use it. If not, then if SSDS has FullSerial, use that. Lacking that, if only a (short) Serial is in BOG, use that instead.
Serial	Shore serial numbers	Copy the BOG short Serial field into a similar field in SSDS.
Model	A real mess	

		Should be as presented and formatted by the manufacturer if possible.

BOG field "Manufacturer"

If at all possible, the Manufacturer should be the complete name of the company, though "Inc" or "Co" can be excluded. (Go to website, how does company refer to itself?) Abbreviations should be expanded except in extreme, well-known cases (e.g., MBARI, WHOI, HOBI Labs).

Until we have a proper company table, we propose to update the company name to the current name whenever it's known to change. (Better answer long term is to create Company Names and Company tables, so multiple names can reference the same company. This can also support lookups of older names and other historical referencing.)

The following table was an initial cut at final names, and is now deprecated. (Kept only for reference.) For the ultimate answers see the final names list.

BOG has...	SSDS Has...	Suggested compromise
Aanderaa	Aanderaa; Aanderaa Instruments	Aanderaa Data Instruments
Asimet	Asimet; ASIMET; Asimet/WHOI; WHOI	WHOI
Benthos	n/a	Benthos
Biospherical	Biospherical	Biospherical
Garmin	Garmin	Garmin
HOBI Labs	Hobi Labs; HOBI Labs, Inc	HOBI Labs
MBARI	MBARI	MBARI
RD Instruments	RDI; RD Instruments	RD Instruments
Satlantic	Satlantic	Satlantic
Sea-Bird	Seabird; Sea-bird; Seabird Electronics; Sea-Bird Electronics; Sea-Bird Electronics Inc	Sea-Bird Electronics
SeaTech	n/a	SeaTech
Simrad	n/a	Simrad
WET Labs	WETLabs; Wetlabs; WET Labs	WET Labs
Xantrex	Xantrex	Xantrex

BOG field "Type"

BOG has...	SSDS has...	Suggested compromise (types	Comments?

		new to SSDS in *italics*)	
ADCP	ADCP	Current Sensor-ADCP	ADCP should have been deprecated in SSDS. It is a particular kind of this type of instrument.
backscatter	Scatterometer; Backscatterometer	HOBI HS2 is a Backscatterometer-Fluorometer; WET Labs BBSB is a Backscatterometer; WET Labs VSFS is a Backscatterometer	HOBI Labs HS2 units are marked as both in SSDS. WETLabs BBSB units are only Scatterometers. Are the two categories distinct or not? In BOG they're both "backscatter"
backscatter/fluorometer	(need to add)	*Backscatterometer-Fluorometer*	In BOG there as WET Labs ECO BB2F-067 and -065
battery	n/a	*Battery*	RDI battery packs
controller; Controller	?	Controller for HydroDAS units; Controller-Mooring Node for OASIS units	Includes MBARI OASIS-x and HOBI HydroDAS
CTD	CTD	CTD	
e-meter	Electrical Sensor	Electrical Sensor	
fluor/turbidity	Fluorometer; Fluorometer-Nephalometer (some ECO-FLNT units marked as Fluormeter)	Fluorometer-Nephelometer	WET Labs ECO-FLNTUS. Some ECO-FLNTU units are incorrectly identified at Fluorometers?). Also, apparently (at least according to Google Fight), it's a nephelometer, not a nephalometer
fluorometer	Fluorometer	Fluorometer	WET Labs ECO FLS and WETStar
GPS	GPS	GPS	
ICC	n/a	*Inductive Modem Cable Coupler*	Not really a distinct component, more like a modem accessory. However, Paul is tracking it in BOG (though he may not in the future).
Meteorological	Metsys	Meteorology Package	Although "Metsys" is used as the common parlance for this package, it is not self-explanatory and is not trivial to derive from a Google search.

modem	Inductive Modem; Inductive Modem-Surface; Inductive Modem-Underwater	Inductive Modem; Inductive Modem-Surface; Inductive Modem-Underwater	Backport from SSDS to BOG. "Inductive Modem" isn't a used category (yet).
Nitrate; Nitrate analyzer	Nitrogen Sensor-ISUS	Nitrogen Sensor-ISUS for MBARI units, Nitrogen Sensor for Satlantic units	Apparently the ISUS and Satlantic units are sufficently distinct that they should be considered different things.
Oxygen optode; Oxygen Optode	Oxygen; Oxygen Sensor	Oxygen Sensor	
pCO2	CO2 Monitor	CO2 Monitor	
Platform	Mooring	Mooring	
power source	?	Power Supply	MBARI WH-CPS-I and WH-CPS-E. This is a unit developed inhouse for powering the Workhorse ADCPs off mooring power
pump	n/a	*Pump*	
radiometer	Radiometer; Radiometer-Hyperspectral; Radiometer-Longwave; Radiometer-Multispectral; Radiometer-Shortwave	Radiometer; Radiometer-Hyperspectral; Radiometer-Longwave; Radiometer-Multispectral; Radiometer-Shortwave	Backport info to BOG
release	n/a	*Release*	
shutter	n/a	Shutter-Antifouling	New term? Replace Bioshutter in SSDS?
temperature/humidity	Meteorology Package	Meteorology Package	Asimet HRH packages
Transducer	n/a	*Transponder*	One instance, Benthos UAT-376. It's a pinger/ transponder.
transmissometer	n/a	*Transmissometer*	Can't find matching instruments in SSDS: SeaTech and WET Labs C-Star units)

BOG field Technician

This field is entirely filled with 'pc' in BOG, and appears to have the same function as the PersonID_FK field in SSDS. (This is not the 'owner' or 'custodian' functions of BOG, but is just the person who knows about this instrument and its properties.) Suggest replacing Paul Coenen as the PersonID_FK for any device in Instru.

Deferred to phase 2:

Calibration Organization	Aandreaa, Benthos, Biospherical, HOBI, HOBI Labs, MBARI, MBARI-Heller, MBARI-Kocher, RD Instruments, RDI, Satlantic, SBE, Sea-Bird, Sea-Bird Electronics, WET Labs, WETLabs,	(line up organization with Manufacturer fields, add person field)
Custodian	_some multiples i.e. _ Coenen/ Heller. Also "??"	
Owner	Mix of people and projects, some with slashes	

Design

This page last changed on Oct 10, 2007 by kgomes.

After the use cases were defined, several activity diagrams were created to help detail out the steps that would need to be taken to implement components to satisify the requirements of the system. Some of the pieces existed already between SSDS and BOG, but the new steps for the different use cases are as follows. Each of the DB changes listed below will cause a write to the database's history table, as indicated in the final scenario.

Current Design details

1. Create New Device Through Instru
    a. Changes are verified (must ensure unique alternate key triple exists)
    b. New entry is inserted into the Device table (set 'show in Instru view' flag to true)
2. Edit Device Through Instru
    a. Changes are verified (must preserve uniqueness of alternate keys)
    b. Changes are made to Device table
3. Create New Device Through SSDS
    a. Need a new Create Device page in SSDS web application
    b. Insert in device table
4. Edit Device Through SSDS
    a. Need to create new Device Edit page (with JSF validation, DAO submission)
    b. Update on Device table

So based on these steps the following components need to be built:

1. Augment Device table per the following:
    a. Create column [MBARI_ID] [int] (but did not set NOT NULL or identity yet)
    b. Create column [Serial] [nvarchar] (50)
    c. Create column [Calibration_organization] [nvarchar] (50)
    d. Create column [Features] [nvarchar] (50)
    e. Create column [Pressure_sensor] [nvarchar] (50)
    f. Create column [Depth_Rating] [int]
    g. Create column [Firmware_EPROM] [nvarchar] (50)
    h. Create column [Memory] [nvarchar] (50)
    i. Create column [Receive_frequency] [nvarchar] (50)
    j. Create column [Transmit_frequency] [nvarchar] (50)
    k. Create column [Enable_code] [nvarchar] (50)
    l. Create column [Release_code] [nvarchar] (50)
    m. Create column [Tilt_option] [nvarchar] (50)
    n. Create column [Purchased_for] [nvarchar] (50)
    o. Create column [Owner] [nvarchar] (50)
    p. Create column [Custodian] [nvarchar] (50)
    q. Create column [Date_new] [datetime]
    r. Create column [PO] [nvarchar] (50)
    s. Create column [Transaction_col] [nvarchar] (10)
    t. Create column [Permanent_comment] [nvarchar] (400)
    u. Create column [document_dir] [nvarchar] (10)
    v. Create column [osg_view] [bit]
    w. Set default on osg_view to '0'

    > ⚠ I changed the column type from ntext to nvarchar for 'Transaction_col', 'Permanent_comment', and 'document_dir' as ntext prevent triggers from being created. Also I shrunk them all down as I did not want to hit the 8060 size limit per row in SQL server. I made 'Transaction_col' and 'document_dir' really small (10) as they have only NULLs in the production database currently. I sent an email to Paul asking about those columns. He stated that the Transaction column was not really used and could be removed.

2. Create a view that matches the instru table structure using the following SQL

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[instru]') and
OBJECTPROPERTY(id, N'IsView') = 1)
```

```
drop view [dbo].[instru]
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

CREATE VIEW dbo.instru
AS
SELECT     ssdsdba.Device.MBARI_ID, ssdsdba.Device.id AS [SSDS ID], ssdsdba.Person.username AS
Technician, ssdsdba.Device.mfgName AS Manufacturer,
ssdsdba.DeviceType.name AS Type, ssdsdba.Device.mfgModel AS Model, ssdsdba.Device.Serial,
ssdsdba.Device.mfgSerialNumber AS FullSerial,
ssdsdba.Device.Calibration_organization AS [Calibration organization],
ssdsdba.Device.Features,
ssdsdba.Device.Pressure_sensor AS [Pressure sensor], ssdsdba.Device.Depth_Rating AS [Depth
Rating],
ssdsdba.Device.Firmware_EPROM AS [Firmware/EPROM], ssdsdba.Device.Memory,
ssdsdba.Device.Receive_frequency AS [Receive frequency],
ssdsdba.Device.Transmit_frequency AS [Transmit frequency], ssdsdba.Device.Enable_code AS
[Enable code],
ssdsdba.Device.Release_code AS [Release code], ssdsdba.Device.Tilt_option AS [Tilt option],
ssdsdba.Device.Purchased_for AS [Purchased for],
ssdsdba.Device.Owner, ssdsdba.Device.Custodian, ssdsdba.Device.Date_new AS [Date new],
ssdsdba.Device.PO,
ssdsdba.Device.Transaction_col AS [Transaction], ssdsdba.Device.Permanent_comment AS
[Permanent comment],
ssdsdba.Device.infoUrlList AS [manufacture web page], ssdsdba.Device.document_dir AS [document
dir]
FROM         ssdsdba.Device LEFT OUTER JOIN
ssdsdba.DeviceType ON ssdsdba.Device.DeviceTypeID_FK = ssdsdba.DeviceType.id LEFT OUTER JOIN
ssdsdba.Person ON ssdsdba.Device.PersonID_FK = ssdsdba.Person.id
WHERE     (ssdsdba.Device.osg_view = 1)

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

3. Use Enterprise Manager to run a DTS job to copy the trans table from Solstice->BOG to SSDS_Metadata (with data)
4. Perform a data normalization step. This will be a bit tricky.
5. Using SQL Query Analyzer, create a foreign key between trans and instru using:

```
ALTER TABLE [dbo].[trans] ADD
CONSTRAINT [FK_trans_instru] FOREIGN KEY
(
[MBARI ID]
) REFERENCES [ssdsdba].[Device] (
[MBARI_ID]
)
GO
```

> **NEED MBARI_IDs first**
>
> This is a bit nasty as we would like to create a foreign key to the view, but we cannot so we should link to MBARI_ID in Device table, but since not all instruments are ones that OSG cares about, we don't have MBARI_IDs for all of them. So in the data normalization step, we need to make sure all entries have MBARI_IDs and the ones from the current instru table match those in the trans table. If we get rid of MBARI_IDs and change them to SSDS_IDs, we will have a different issue.

6. Create a System DSN on the machine where instrumentsDE will run and point to the SSDS_Metadata database but call it "BOG". Use the ssdsdba login and password.

7. When you open up the instrumentsDE, you will need to create new linked tables for instru and trans that point to the new DSN and the view and table in SSDS_Metadata.
8. Make changes to the instrumentsDE application
9. Create a history table for the Device table using the stored procedure:

```
EXEC AdminGenerateHistoryTable 'Device'
go
```

> **Make sure not History_Device table exists**
>
> In order for the stored procedure to work correct drop any History_Device table that may exist

10. Create the triggers using the stored procedure:

```
EXEC AdminGenerateHistoryTriggers 'Device'
go
```

> I found that I had to remove the old triggers before running this.

11. Drop both history tables for DeviceType and Person
12. Remove all triggers on DeviceType and Person tables
13. Generate History tables for DeviceType and Person

```
EXEC AdminGenerateHistoryTable 'DeviceType'
go

EXEC AdminGenerateHistoryTable 'Person'
go
```

14. Generate Triggers for DeviceType and Person

```
EXEC AdminGenerateHistoryTriggers 'DeviceType'
go

EXEC AdminGenerateHistoryTriggers 'Person'
go
```

> OK, so now that all that is in place, we have a bit of a problem with updates of the columns 'Technician' and 'Type' in the access application. When 'Technician' is edited the Person.username will be changed. When the 'Type' is edited the DeviceType.name will be changed. This is where INSTEAD OF triggers should save our behind.

15. The history tables store the updated values for entries as the thought is that each entry would have to be inserted before any updates so you would always have the initial state of the row. In this case we are starting with existing data, so it is critical that we get a baseline of all the current rows. This can be done simply by running these in SQL Query Analyzer:

```
UPDATE ssdsdba.Device SET version = version
GO

UPDATE ssdsdba.Person SET version = version
GO

UPDATE ssdsdba.DeviceType SET version = version
GO
```

    This will create entries for all the rows so that we will have the current data in the history table to compare future updates/deletions against.
16. Create an INSTEAD OF INSERT trigger on the view

```
CREATE TRIGGER trInstruInsert ON instru
INSTEAD OF INSERT
AS
BEGIN
-- Check to see if there are updated rows
IF EXISTS (Select * from Inserted)
```

```
BEGIN
-- Declare any needed variables
DECLARE @mbari_id int,
@ssds_id numeric(9),
@username varchar(50),
@mfgName varchar(255),
@deviceTypeName varchar(255),
@mfgModel varchar(255),
@serial varchar(50),
@mfgSerialNumber varchar(255),
@calibration_organization nvarchar(50),
@features nvarchar(50),
@pressure_sensor nvarchar(50),
@depth_rating int,
@firmware_eprom nvarchar(50),
@memory nvarchar(50),
@receive_frequency nvarchar(50),
@transmit_frequency nvarchar(50),
@enable_code nvarchar(50),
@release_code nvarchar(50),
@tilt_option nvarchar(50),
@purchased_for nvarchar(50),
@owner nvarchar(50),
@custodian nvarchar(50),
@date_new datetime,
@po nvarchar(50),
@transaction_col nvarchar(50),
@permanent_comment nvarchar(50),
@infoUrlList varchar(2048),
@document_dir nvarchar(50),
@person_id numeric(9),
@deviceType_id numeric(9)

-- Now grab all the values from the Inserted table
SELECT
@mbari_id = MBARI_ID,
@ssds_id = [SSDS ID],
@username = Technician,
@mfgName = Manufacturer,
@deviceTypeName = Type,
@mfgModel = Model,
@serial = Serial,
@mfgSerialNumber = FullSerial,
@calibration_organization = [Calibration organization],
@features = Features,
@pressure_sensor = [Pressure sensor],
@depth_rating = [Depth Rating],
@firmware_eprom = [Firmware/EPROM],
@memory = Memory,
@receive_frequency = [Receive Frequency],
@transmit_frequency = [Transmit Frequency],
@enable_code = [Enable code],
@release_code = [Release code],
@tilt_option = [Tilt option],
@purchased_for = [Purchased for],
@owner = Owner,
@custodian = Custodian,
@date_new = [Date new],
@po = PO,
@transaction_col = [Transaction],
@permanent_comment = [Permanent comment],
@infoUrlList = [manufacture web page],
@document_dir = [document dir]
FROM Inserted
-- Now grab the technician name
SELECT @username = Technician FROM Inserted
```

```
-- Now grab the device type name
SELECT @deviceTypeName = Type FROM Inserted

-- Now let's make sure the MBARI_ID and SSDS_ID are null
IF @mbari_id IS NOT NULL
RAISERROR('The insert specified the MBARI_ID. This field is auto-generated, do not specify on
insert',9,1)
IF @ssds_id IS NOT NULL
RAISERROR('The insert specified the SSDS_ID. This field is auto-generated, do not specify on
insert',9,1)

-- Next, let's check to see if the technician's name is not null
IF @username IS NOT NULL
BEGIN
-- Now check to make sure it is just not empty
IF @username != ''
BEGIN
-- Check to see if it exists
IF NOT EXISTS (SELECT id FROM ssdsdba.Person WHERE username = @username)
INSERT INTO ssdsdba.Person (version, username, email) VALUES (0, @username, @username)
SELECT @person_id = id FROM ssdsdba.Person WHERE username = @username
END
ELSE
SET @person_id = NULL
END
ELSE
SET @person_id = NULL

-- Let's now do the same thing for the device type
IF @deviceTypeName IS NOT NULL
BEGIN
-- Now check to make sure it is just not empty
IF @deviceTypeName != ''
BEGIN
-- Check to see if it exists
IF NOT EXISTS (SELECT id FROM ssdsdba.DeviceType WHERE name = @deviceTypeName)
INSERT INTO ssdsdba.DeviceType (version, name) VALUES (0, @deviceTypeName)
SELECT @deviceType_id = id FROM ssdsdba.DeviceType WHERE name = @deviceTypeName
END
ELSE
SET @deviceType_id = NULL
END
ELSE
SET @deviceType_id = NULL

-- First update the columns that map directly
INSERT INTO ssdsdba.Device
(
mfgName,
mfgModel,
Serial,
mfgSerialNumber,
Calibration_organization,
Features,
Pressure_Sensor,
Depth_Rating,
Firmware_EPROM,
Memory,
Receive_frequency,
Transmit_frequency,
Enable_code,
Release_code,
Tilt_option,
Purchased_for,
Owner,
Custodian,
```

```
Date_new,
PO,
Transaction_col,
Permanent_comment,
infoUrlList,
document_dir,
PersonID_FK,
DeviceTypeID_FK
)
VALUES
(
@mfgName,
@mfgModel,
@serial,
@mfgSerialNumber,
@calibration_organization,
@features,
@pressure_sensor,
@depth_rating,
@firmware_eprom,
@memory,
@receive_frequency,
@transmit_frequency,
@enable_code,
@release_code,
@tilt_option,
@purchased_for,
@owner,
@custodian,
@date_new,
@po,
@transaction_col,
@permanent_comment,
@infoUrlList,
@document_dir,
@person_id,
@deviceType_id
)
END
END
GO
```

17. Create an INSTEAD OF UPDATE trigger on the view

```
CREATE TRIGGER trInstruUpdate ON instru
INSTEAD OF UPDATE
AS
BEGIN
-- Check to see if there are updated rows
IF EXISTS (Select * from Inserted)
BEGIN
-- Declare any needed variables
DECLARE @mbari_id int,
@ssds_id numeric(9),
@username varchar(50),
@mfgName varchar(255),
@deviceTypeName varchar(255),
@mfgModel varchar(255),
@serial varchar(50),
@mfgSerialNumber varchar(255),
@calibration_organization nvarchar(50),
@features nvarchar(50),
@pressure_sensor nvarchar(50),
@depth_rating int,
@firmware_eprom nvarchar(50),
@memory nvarchar(50),
@receive_frequency nvarchar(50),
```

```
@transmit_frequency nvarchar(50),
@enable_code nvarchar(50),
@release_code nvarchar(50),
@tilt_option nvarchar(50),
@purchased_for nvarchar(50),
@owner nvarchar(50),
@custodian nvarchar(50),
@date_new datetime,
@po nvarchar(50),
@transaction_col nvarchar(50),
@permanent_comment nvarchar(50),
@infoUrlList varchar(2048),
@document_dir nvarchar(50),
@person_id numeric(9),
@deviceType_id numeric(9)

-- Now grab all the values from the Inserted table
SELECT
@mbari_id = MBARI_ID,
@ssds_id = [SSDS ID],
@username = Technician,
@mfgName = Manufacturer,
@deviceTypeName = Type,
@mfgModel = Model,
@serial = Serial,
@mfgSerialNumber = FullSerial,
@calibration_organization = [Calibration organization],
@features = Features,
@pressure_sensor = [Pressure sensor],
@depth_rating = [Depth Rating],
@firmware_eprom = [Firmware/EPROM],
@memory = Memory,
@receive_frequency = [Receive Frequency],
@transmit_frequency = [Transmit Frequency],
@enable_code = [Enable code],
@release_code = [Release code],
@tilt_option = [Tilt option],
@purchased_for = [Purchased for],
@owner = Owner,
@custodian = Custodian,
@date_new = [Date new],
@po = PO,
@transaction_col = [Transaction],
@permanent_comment = [Permanent comment],
@infoUrlList = [manufacture web page],
@document_dir = [document dir]
FROM Inserted
-- Now grab the technician name
SELECT @username = Technician FROM Inserted
-- Now grab the device type name
SELECT @deviceTypeName = Type FROM Inserted

-- Next, let's check to see if the technician's name is not null
IF @username IS NOT NULL
BEGIN
-- Now check to make sure it is just not empty
IF @username != ''
BEGIN
-- Check to see if it exists
IF NOT EXISTS (SELECT id FROM ssdsdba.Person WHERE username = @username)
INSERT INTO ssdsdba.Person (version, username, email) VALUES (0, @username, @username)
SELECT @person_id = id FROM ssdsdba.Person WHERE username = @username
END
ELSE
SET @person_id = NULL
END
```

```
ELSE
SET @person_id = NULL

-- Let's now do the same thing for the device type
IF @deviceTypeName IS NOT NULL
BEGIN
-- Now check to make sure it is just not empty
IF @deviceTypeName != ''
BEGIN
-- Check to see if it exists
IF NOT EXISTS (SELECT id FROM ssdsdba.DeviceType WHERE name = @deviceTypeName)
INSERT INTO ssdsdba.DeviceType (version, name) VALUES (0, @deviceTypeName)
SELECT @deviceType_id = id FROM ssdsdba.DeviceType WHERE name = @deviceTypeName
END
ELSE
SET @deviceType_id = NULL
END
ELSE
SET @deviceType_id = NULL

-- First update the columns that map directly
UPDATE ssdsdba.Device
SET ssdsdba.Device.mfgName = @mfgName,
ssdsdba.Device.mfgModel = @mfgModel,
ssdsdba.Device.Serial = @serial,
ssdsdba.Device.mfgSerialNumber = @mfgSerialNumber,
ssdsdba.Device.Calibration_organization = @calibration_organization,
ssdsdba.Device.Features = @features,
ssdsdba.Device.Pressure_Sensor = @pressure_sensor,
ssdsdba.Device.Depth_Rating = @depth_rating,
ssdsdba.Device.Firmware_EPROM = @firmware_eprom,
ssdsdba.Device.Memory = @memory,
ssdsdba.Device.Receive_frequency = @receive_frequency,
ssdsdba.Device.Transmit_frequency = @transmit_frequency,
ssdsdba.Device.Enable_code = @enable_code,
ssdsdba.Device.Release_code = @release_code,
ssdsdba.Device.Tilt_option = @tilt_option,
ssdsdba.Device.Purchased_for = @purchased_for,
ssdsdba.Device.Owner = @owner,
ssdsdba.Device.Custodian = @custodian,
ssdsdba.Device.Date_new = @date_new,
ssdsdba.Device.PO = @po,
ssdsdba.Device.Transaction_col = @transaction_col,
ssdsdba.Device.Permanent_comment = @permanent_comment,
ssdsdba.Device.infoUrlList = @infoUrlList,
ssdsdba.Device.document_dir = @document_dir,
ssdsdba.Device.PersonID_FK = @person_id,
ssdsdba.Device.DeviceTypeID_FK = @deviceType_id
WHERE ssdsdba.Device.id = (Select [SSDS ID] from Inserted)
END
END
GO
```

18. Create an INSTEAD OF DELETE trigger on the view

```
CREATE TRIGGER trInstruDelete ON instru
INSTEAD OF DELETE
AS
BEGIN
-- Check to see if there are updated rows
IF EXISTS (Select * from Deleted)
BEGIN
-- Declare any needed variables
DECLARE @mbari_id int,
@ssds_id numeric(9),
@username varchar(50),
@mfgName varchar(255),
```

```
@deviceTypeName varchar(255),
@mfgModel varchar(255),
@serial varchar(50),
@mfgSerialNumber varchar(255),
@calibration_organization nvarchar(50),
@features nvarchar(50),
@pressure_sensor nvarchar(50),
@depth_rating int,
@firmware_eprom nvarchar(50),
@memory nvarchar(50),
@receive_frequency nvarchar(50),
@transmit_frequency nvarchar(50),
@enable_code nvarchar(50),
@release_code nvarchar(50),
@tilt_option nvarchar(50),
@purchased_for nvarchar(50),
@owner nvarchar(50),
@custodian nvarchar(50),
@date_new datetime,
@po nvarchar(50),
@transaction_col nvarchar(50),
@permanent_comment nvarchar(50),
@infoUrlList varchar(2048),
@document_dir nvarchar(50),
@person_id numeric(9),
@deviceType_id numeric(9)

-- Now grab all the values from the Deleted table
SELECT
@mbari_id = MBARI_ID,
@ssds_id = [SSDS ID],
@username = Technician,
@mfgName = Manufacturer,
@deviceTypeName = Type,
@mfgModel = Model,
@serial = Serial,
@mfgSerialNumber = FullSerial,
@calibration_organization = [Calibration organization],
@features = Features,
@pressure_sensor = [Pressure sensor],
@depth_rating = [Depth Rating],
@firmware_eprom = [Firmware/EPROM],
@memory = Memory,
@receive_frequency = [Receive Frequency],
@transmit_frequency = [Transmit Frequency],
@enable_code = [Enable code],
@release_code = [Release code],
@tilt_option = [Tilt option],
@purchased_for = [Purchased for],
@owner = Owner,
@custodian = Custodian,
@date_new = [Date new],
@po = PO,
@transaction_col = [Transaction],
@permanent_comment = [Permanent comment],
@infoUrlList = [manufacture web page],
@document_dir = [document dir]
FROM Deleted

-- Now let's make sure the SSDS_ID is not null
IF @ssds_id IS NULL
RAISERROR('The delete did not specify the SSDS_ID. No delete performed',9,1)

-- Now delete the row specified
DELETE FROM ssdsdba.Device
WHERE id = @ssds_id
```

```
END
END
GO
```

19. Person Creation/Edit/Delete page (KG)
20. DeviceType Creation/Edit page (KG)
21. Device Creation Page and Device Edit Page (KG)
    a. We will need a unique device edit page that can edit non-SSDS fields (like OSG view flag).
    b. Authenticate against LDAP group
    c. JSF Validation (unique alternate keys; what else?)
    d. Calls DAO to create new device/submit changes to existing device
22. External application (JG/AM)
    a. Runs on interval (5 minutes) against history table
    b. Checks for new devices
        i. mail to SSDS admin with key and alternate key fields (all fields?), and whether device is exposed through Instru view
        ii. send mail to OSG admin (with link to kick of the servlet to make it visible)...
            i. only if not visible in Instru? (Does OSG want notification on *all* additions?)
            ii. only if not OSG-created? (If OSG creates thru SSDS, send mail? set Instru-visible flag?)
        iii. If OSG operator clicks link, servlet is called that sets Instru view flag on that device
    c. Checks for edited devices
        i. mail to SSDS admin with changed fields (old and new), who and when changed, whether device is exposed through Instru view
        ii. if exposed through Instru view, send same mail to OSG admin with link to edit
            i. just point back to edit the device for now; later can make this a link to revert the change
            ii. does OSG admin want to be notified of changes by OSG admin personnel?
    d. Marks each checked item as done
23. 1 Servlet (KG)
    a. Change Instru view flag
        i. Pass in SSDS Device ID
        ii. Set the flag on that device to allow it to show through the Instru view
        iii. Send email to SSDS Admin confirming change
24. 2 History Triggers on Device Table (KG)
    a. Every time a device is added, a trigger writes to a history table
        - Information includes key, the alternate key fields, who added the device, and when
    b. Every time a device is edited, a trigger writes to a history table
        - Information includes key, old record, new record (or old field, new field if atomic), who, when
25. SQL to perform initial database normalization between BOG and SSDS (JG/AM)
    a. All XML in puckxml will have to have matching changes made
26. Custom field comparison utilities
    a. Serial number
        i. Ignore non-alphanumeric text (remove white space, punctuation)
        ii. If existing serial number matches the end of the new serial number, consider it a (likely?) match
    b. Model number
        i. Ignore non-alphanumeric text (remove white space, punctuation)
27. Put links to CVS web xml in Device listing web page (KG)

Notes:

A proposal for an integrated set of database fields: DatabaseFields

AccessAppChanges

During our design, questions were raised:

1. Does the Microsoft Access application have any validation (check for duplicate serial numbers, etc.)?

Generally speaking, no. Some fields (Mfg, Model, Device, ...)are regularized by using drop-downs on the new/edit dialog boxes.

AccessAppChanges

This page last changed on Feb 22, 2007 by kgomes.

I'll use the term "OSG-owned" to indicate the device Paul cares about. We haven't exactly figured out how this will be delimited in the database, but I assume we'll be able to SELECT those lines as part of the SQL query.

Summary

- Remove all references to MBARI_ID (auto-increment key used in instru table) and replace with SSDS_ID (some sort of key from the merged table). Note: Access assumes a relatively short MBARI_ID (4 digits), not the full SSDS UUID.

  Are we sure about this?

  I am not sure we need to necessarily get rid of the MBARI_ID. Yes, we would have two auto-incrementing fields, but that may not be all that big of a deal and the Access application expects an MBARI_ID and that would minimize our code changes to the Access app.

  We need to make it so that the SSDS_ID is not editable in the Access application

- Several panes in the Access app use a series of drop-downs (Manufacturer, Model, Serial Number, Type, etc) to select one-or-more devices. Whenever possible, an "OSG-owned" checkbox could be added to limit/unlimit the search.

  Done through view

  In theory, this should be accomplished by presenting a DB view that is keyed on the OSG view flag in the database.

  Now that the Technician and Type are mapped through a view to the Person.username and DeviceType.name fields, those become editable in the current access application. It would be better if these could be specified as drop-downs always and then were edited through the SSDS appropriate pages.

- There are a few panels which display all of the fields of the instru database, and a few which also allow editing of those fields. These need a clear and concise way of indicating and editing the "OSG-ownership"

  Also done through view

  In theory, this should be accomplished by presenting a DB view that is keyed on the OSG view flag in the database.

  Overall impact to the Access app would be surprisingly light. The only truly obstrusive change would be to the "Instrumentation Data Entry" pane which is primarily for new instrument entry. At present it's just a flat pane with fields for data entry (some drop-downs to normalize answers), but we're proposing changing it to a two-step process where some key information is entered, then checked for an existing records, before proceeding.

There will also be the introduction of a number of checkboxes which select if a particular search should be limited OSG-owned devices (the default) or expanded to the full SSDS database.

I think we can ignore this

I think we can safely ignore this and just assume that OSG will only want to search/work on entries that have the OSG flag enabled and we will just use the view for that. If they

want to expose the device through to the OSG app, they will have to use another interface to flip the bit to make it visible.

There's likely to be more impacts in changes to the meaning and function of some of the database fields.

There are also some loose ends in data normalization and field mapping (i.e. ensuring the FullSerial is always filled out, even if the user has only entered a Serial). That sort of data verification and manipulation could be encoded in Access or pushed off to the SQL server as a trigger/job.

More detail

From the front index page, there are nine sub-pages:

Transactions Data Entry

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections
- I'm a little confused by the usage of this pane – it's a combination of some instrumentation entries and the transaction entries.

Instruments Data Entry

- Replace MBARI_ID with SSDS_ID.
- Add a way to indicate "OSG-owned"
- Need to put in an explicit step where the first few key fields are filled out and the database is check for duplicates before filling out the remaining fields. This function should probably ignore the "OSG-owned" field. This window may expand to include both the add and edit functions.

Current Info

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections
- Replace MBARI_ID with SSDS_ID

Transaction History

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections
- Replace MBARI_ID with SSDS_ID

Find Location of Calibration

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections
- Replace MBARI_ID with SSDS_ID (takes information from trans table?)

Query by Location

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections
- Replace MBARI_ID with SSDS_ID on list.
- Selecting "details" brings up a tertiary window. Remove MBARI_ID. Add entry on window for "OSG-owned" (may be same detail window used in "Find Model")

Find Model

- Add checkbox to filter list on "OSG-owned" – affects drop-down selections.
- Replace MBARI_ID with SSDS_ID on list
- Selecting "details" brings up a tertiary window. Remove MBARI_ID. Add entry on window for "OSG-owned"

Deployment Update

(this feature may not be functional or as functional as Paul would like.)

- At checkbox to filter list on "OSG-owned" – affects list of instruments
- Change MBARI_ID to SSDS_ID 

Edit Instruments and Transactions Table

- Add checkbox in top pane to filter out OSG-owned devices – affects drop-down selections.
- Add entry box in lower pane to toggle "OSG-owned"
- Remove references to MBARI_ID in text boxes and transaction table. Replace with SSDS ID.

DatabaseFields

This page last changed on Feb 26, 2007 by graybeal.

Here's a proposal for a single database which is a union of the existing SSDS and Instru databases. It should be possible to use a simple d/b view to translate this table to either the existing SSDS or instru formats without too many machinations. The goal is to minimize/eliminate redundancy by combining overlapping fields without drastically affecting the use of either of the existing apps (SSDS, Access app).

The steps to actually create the data in these fields are documented in MakingData.

This should be considered a phase one proposal to cover most of the "low hanging fruit" and start the integration process. Subsequent phases can further integrate the two databases while taking into consideration lessons learned from phase one.

A good example is the handling of people/roles. The instru database has a number of separate fields for different roles: owner/purchaser, technician, calibrator, etc. Each of these fields is just freetext, allowing for inclusion of multiple people for each role, also project names instead of peoples names. The SSDS database recognizes a single person, which is a record of type Person. Though it's not clear how this record was intended to be used, it does not map cleanly into the instru model.

Despite this data mismatch, it isn't strictly necessary to solve this conundrum for successful use of the integrated database. Later phases may include revisions to either or both data models (for example use of an associative array between ISO roles and Person records in SSDS).

Given the ability to "alias" fields when creating a view, name changes or mismatches should not be considered particularly serious.

Pending:

- Calibration information? This is a new addition and not really relevant to the "merge"
- should probably settle on a word formatting sheme (i.e. starts with lower case and capitalize to make subwords) – given we can alias when making the view it matters quite a bit less

SSDS Field	SSDS Comment	instru Field	instru comment	Merged field	Merge comment
SSDS ID	Auto-incrementing key field	SSDS ID	Used to link to entry in SSDS d/b	SSDS ID	
UUID				UUID	
mfgName		Manufacturer		mfgName	Needs some normalization
mfgModel		Model		mfgModel	Needs some normalization
mfgSerialNumber		Serial, FullSerial	One or both may be used. Typically FullSerial is the complete manufacturer's serial number and Serial is a shorter representation - often the actual sequential portion w/ o any sort	mfgSerialNumber, ShortSerial	Keep *mfgSerialNumber* as the full-length serial number. Make it a mandatory field. Allow an optional *ShortSerial* to hold a shortened serial number, which is a copy of the Instru Serial field.

			of model identification which may be encoded.		
name	Typically something like manufacturer +model			name	
deviceType		Type		deviceType	Needs some normalization
Description	A free text description, may contain some information redundant with *name*			Description	
infoUrlList		manufacture web page		infoUrlList	Neither database uses this field very heavily. Best to merge into a single "place for a hyperlink" – though should we be specific that this is to either the manufacturers' home page or to the product page. Does it matter?
person	The contact person (link to the Person table)			person	Need to figure out how (if) to map the multiple instru roles into SSDS
		Purchased for	Sometimes a person, sometimes a project	Purchased for	"
		Owner	Typically some combination of person, group (OSG, etc) and project. Sometimes all three	Owner	"
		Custodian	One or more people	Custodian	"
		Technician	Currently always	Technician	"

			'pc' (Paul Coenen)		
resources	Not used currently (?). Could be used for storing calibration files?				
		document dir	Not used currently		
version				version	
		Features	A free text description of features on the device	Features	Simplest to just leave this in place for now?
		Pressure sensor	Just freetext, left blank if irrelevant	Pressure sensor	"
		Depth Rating	Just freetext, left blank if irrelevant	Depth Rating	"
		Firmware/ EPROM	Just freetext, left blank if irrelevant	Firmware/ EPROM	"
		Memory	Just freetext, left blank if irrelevant	Memory	"
		Receive Frequency	Just freetext, left blank if irrelevant	Receive Frequency	"
		Transmit Frequency	Just freetext, left blank if irrelevant	Transmit Frequency	"
		Enable Code	Just freetext, left blank if irrelevant	Enable Code	"
		Release Code	Just freetext, left blank if irrelevant	Release Code	"
		Tilt option	Just freetext, left blank if irrelevant, though heavier use of "N/A" than in other fields	Tilt option	"
		Date new	Used infrequently	Date new	"

		Calibration organization	Typically a company, but sometimes a company and a person	Calibration organization	In the long run we might normalize this against the msgNames
		PO		PO	"
		Permanent comment	Not entirely sure what this field is used for. I think it's meant to be some permanent and overarching state of the system, not just the latest "transaction"	??	Keep it? Or discard?
		MBARI_ID	Auto-incrementing key field	*(discard)*	Redundant if the **trans** table in instru is rewritten using SSDS IDs instead
		Transaction	I believe this field isn't used in instru	*(discard)*	

MakingData

This page last changed on Mar 14, 2007 by graybeal.

This document contains the steps needed to generate the data in BogInstruNormalization and DatabaseFields.

Assumptions

The scripts make the following assumptions:

1. The Device table has had columns added to match the list under DeviceMerge.
2. An up-to-date copy of the Instru table is in the SQL Server where these changes will be made.
    - It is OK to change the copy of the Instru table.
    - Must make copy of this table as a different name (Instru is name of view)
3. The person running the script has permissions to:
    - Change Device.
    - Change Instru.
    - Create new tables and leave them (Manufacturers) or delete them later.
4. Either DeviceTypes has been updated to reflect the changes in Type, or the SQL at SqlQueriesFinal#AddDeviceTypes is run to make the necessary changes.

In some cases the steps are 'smart' about using existing data to assess changes, but in other cases it was necessary to assign the values directly for some of the devices. SSDS ID is used as the key for most of these changes; barring unusual changes this should not break.

I made the following assumptions:

1. Paul C approves of all these changes.
2. Someone will quickly look over the code to see if there may be bad side effects. I was careful but I'm not that skillful yet.

The Columns

We start from the following list of columns in the dbo.Instru view, corresponding to SSDS columns as noted (proposed changes in *italics*):

Instru View	SSDS Field if diff
MBARI ID	
SSDS-ID	id
Technician	*PersonID_FK*
Manufacturer	mfgName
Type	name
Model	mfgModel
Serial	
FullSerial	mfgSerialNumber
Calibration-organization	
Features	
Pressure-sensor	
Depth-Rating	

Firmware/EPROM	
Memory	
Receive-frequency	
Transmit-frequency	
Enable-code	
Release-code	
Tilt-option	
Purchased-for	
Owner	
Custodian	
Date-new	
PO	
Transaction	
Permanent-comment	
manufacture-web-page	infoUrlList
document-dir	

Initial Normalization of Devices

The first step is to ensure we know which of the items in the BOG Instru database that don't have SSDS_IDs are also in the SSDS database. (It is important that we minimize the number of duplicate entries as of the end of the process, so we might as well do this step first.) This query must first find BOG devices that could be already entered as SSDS devices. To do this, we look for possibly matching serial numbers where there is no meaningful SSDS_ID in the Instru database. We can then cross-check the models and instrument types manually to see if it is likely to be the same instrument.

When in doubt, we should assume the two instruments are different, until it is clear that they are the same.

The following instruments are believed the same. (Queries are at SqlQueries#FindMatchingDevices.) This will be reflected by setting their SSDS ID in Instru to the corresponding SSDS ID.

Instru	SSDS	Comment
10	1359	Full serial numbers match
23	1567	
50	1568	
129	1222	
162	1273	
195	1337	

201	1279	
204	1356	
227	1470	
288	1562	
51	1382	Short serial numbers match
137	1406	
157	1267	
175	1456	
176	1457	
190	1314	
256	1459	

Implement this by doing updates on the SSDS ID values of the appropriate entries in the instru table. See SqlQueriesFinal#MatchDevices.

Device Merge (Merging Instru Table into Device Table)

The next step is to migrate most of the Instru data 'as is' into the Device Table. This will have two aspects: the devices without SSDS IDs, and the devices with SSDS IDs. In each case, only the fields which did not already exist in SSDS will be migrated. The remaining fields will be accomplished in separate steps below. (An exception is the PersonID_FK field, which is easiest to do as part of this step.)

This step is performed now so that we have a single table with SSDS IDs for every item, which will be useful for future queries.

For this step, we assume the Device table already exists with the new Instru fields required for the merge. Those fields are:

MBARI_ID
Serial
FullSerial (mfgSerialNumber)
Calibration-organization
Features
Pressure-sensor
Depth-Rating
Firmware/EPROM
Memory
Receive-frequency
Transmit-frequency
Enable-code
Release-code
Tilt-option
Purchased-for
Owner
Custodian
Date-new
PO
Transaction
Permanent-comment
document-dir

Devices without SSDS IDs

Create new devices in Device, copying all the new fields from Instru into the corresponding columns of Device.

- manufacturer-web-page gets copied to infoUrl.

Also set the PersonID_FK = 126 (for Paul Coenen – see Technician below).

Then copy the SSDS ID for the new devices back into the SSDS ID column of Instru, to make later SQL simpler.

Use this Detailed SQL for everything in this section.

Devices with SSDS IDs

Update the new Instru fields in Device with the data from Instru.

- manufacturer-web-page gets copied to infoUrl.
  '

  Use this Detailed SQL.

Back to table...

Technician

For any device in Instru, set PersonID_FK in SSDS to 126 (Paul Coenen). (See explanation in BogInstruNormalization.)

Strategy: This is most easily accomplished during the device merge step.
(Note, by the way, there are 15 additional items in Device with PersonID_FK = coenen that aren't already in Instru.)

Originally we planned to change all the 'pc' values in Instru, as shown in SqlQueries#ChangeTechnician. I don't believe this step will be necessary.

MBARI_ID and SSDS ID

For reasons explained in the Design page, we must support MBARI ID field in the BOG database. To ensure proper operations of both systems, we want to guarantee that all devices have both an MBARI ID and an SSDS ID. To do this, as we merge the two systems we will generate any missing IDs, according to the following rules:

- SSDS ID is set for new Instru devices as they are added to the database. This happens as part of the device merge above.
- If MBARI ID doesn't exist, it is set to the SSDS ID as part of the device merge.
- If the MBARI ID already exists, it will need to be changed to match SSDS. (This can happen after everything else is done.)

- ***if we need to save the Instru MBARI ID for some reason, this can be moved later; else do it as part of device merge.***
- prepare any associated changes to the FK in Trans.
- Set the MBARI ID in Device to the SSDS ID.
- Will also have to set appropriate constraints on MBARI_ID column

In the future, any new device receives the next available SSDS ID, and the MBARI ID is set to the same number.

Interim Table Creation

Next we will create a table to use for interim results. (This table will be reused in subsequent steps.) The table will have an entry for each of the devices in SSDS – note that some of these will be in Device only,

and the rest will be in both tables (some devices that were originally Instru-only, and others that were originally in both tables). All subsequent queries have to allow for all 3 possibilities, until all the Instru information has been migrated.

SQL for this task will be in SqlQueriesFinal#MakeInterimTable.

Back to table...

Manufacturer

(See normalization strategy.)

The list of permitted manufacturers (so far, based on existing data as of 2/20) is as follows. The name in parentheses provides the complete company name, in case we ever make a database.

- Aanderaa Data Instruments
- Axys Technologies (.. Inc.) (was Axys Environmental Systems)
- Biospherical Instruments (.. Inc.)
- Bluefin Robotics (.. Corporation)
- Brook Ocean Technology (.. Limited)
- Crossbow Technology (.. Inc.)
- Garmin
- HOBI Labs (Hydro-Optics, Biology, & Instrumentation Laboratories)
- Magellan Navigation (.., Inc.)
- MBARI (Monterey Bay Aquarium Research Institute)
- McLane Research Laboratories (.., Inc.)
- NOBSKA
- ORBCOMM (.., Inc.)
- Paroscientific (.., Inc.)
- QUAKE Global (.., Inc.)
- Satlantic (.. Inc.)
- Sea-Bird Electronics (.., Inc.)
- SeaTech (.., Inc.) (no longer exists)
- Simrad (.., Inc)
- Star Engineering
- Teledyne Benthos (was Benthos)
- Teledyne RD Instruments (was RD Instruments)
- UC Santa Barbara (University of California, Santa Barbara)
- WET Labs
- WHOI (Woods Hole Oceanographic Institution)
- Xantrex Technology (.. Inc.)

The following names are also supported

- unknown
- tbd
- testing

The algorithm to follow for the merge will be:

1. If the name in SSDS or Instru matches one in the above lists, use it.
2. If the first 3 letters of the name in SSDS or Instru is contained in (ignoring case)one of the names above, use the name it is contained in.
    - Note that SeaTech entries should have been an exact match in step 1; all others map to Sea-Bird Electronics.
3. Perform manual corrections listed below.

The following entries require manual correction (starred entries should be OK'd with Paul C):

1. Anderraa Instruments -> Aanderaa Instruments (typo fix)
2. * Ashtech -> Magellan Navigation (Ashtech is model not company) (let Paul know)
3. * ASIMET -> Star Engineering (model not company; allegedly Star Eng is only seller)
4. * Asimet/WHOI -> Star Engineering (model not company; allegedly Star Eng is only seller)
5. * Triaxys -> Axys Technologies
6. * UCSB -> UC Santa Barbara

These entries shld be addressed by the 'contained in' rule above.

1. Axys Environmental Systems -> Axys Technologies (name change?)
2. * Benthos -> Teledyne Benthos (bought))
3. Biospherical -> Biospherical Instruments
4. Brook Ocean Technology Ltd. -> Brooke Ocean Technology
5. Nobska -> NOBSKA
6. OrbComm -> ORBCOMM
7. Quake -> QUAKE Global
8. * RD Instruments -> Teledyne RD Instruments (bought)
9. Xantrex -> Xantrex Technology

The plan

To do this, as a first step create a table of allowable Manufacturers (see SqlQueriesFinal#CreateMfgTable). The label column is the name that will be used to fill out the mfgName field in Device. (Conceivably the manufacturers information in Device should all be relational, but for now this is a quick and dirty.)

Then we will copy the appropriate information over to our interim table (TempDevFld) using the mfgName column, from Instru or Device as appropriate, using the rules above.

At this point, all the mfgName rows should be filled in.

Manual changes will be done to the interim table.

Finally, the InterimTable will be used to overwrite the Device table, and the workingText column of Interim table will be cleared.

This is shown in SqlQueriesFinal#UpdateMfgName.

Back to table...

Model

(See normalization strategy.)

The following names are also supported

- unknown
- none
- tbd
- testing

At some point (phase II?), figure out issues with:

1. SSDS id 1009, 1218 (2 'MTM' with serial=1)
2. SSDS id 1216 (GPS, no model)
3. Are Metsys 1321, 1396, 1397 really MBARI mfg?
4. What to do with virtual hardware (1309-1311)?
5. delete test entries? (1509 etc.)
6. SideARM (1278, 1295, 1300, 1411, 1413) vs MSP
7. SSDS id 1275 (MBARI Gashound)
8. Are all Aquadopp entries the same units (mfgModel is long and identical but descriptions differ)
9. SSDS id 1323 (is it a SOON, is it a pco2, is it a platform?)

Before beginning, confirm with Paul that the following changes are OK:

1. all Sea-Bird CTDs have model numbers of the form SBE nn[-IMP]
    - SBE prefix
    - '-' after number, except for 'plus' versions
2. changes listed under manual correction below

The following entries require 'manual' correction (assume Paul has OK'd the above):

1. Make the models for the following SSDS ids as listed:

    - 1313: GPS16-HVS
    - 1395: ECO FLNTUSB

1.
    - all '(null)' to null
    - all 'ECO-FLNT*' to 'ECO FLNT*'
    - all 'triplet' to 'ECO Triplet'
    - all 'MMC v*' to 'MMCv4'
    - all 'Medusa R*' to 'Medusa Rev 2'
    - all 'Medusa v*' to 'Medusa Rev 2'
    - all 'Fiberglass tor*' to 'OASIS Buoy'
    - all 'Emeter' to 'E-meter'

The algorithm to follow for the merge (order matters) will be:

1. If BOG Instru Model exists use that; else use SSDS mfgModel. This step is accomplished in DeviceMerge.
2. Do manual corrections as listed above (code in SqlQueriesFinal#ManualModel).

Back to table...

Serial

For the merge: Copy the contents of this field without changes.

Back to table...

FullSerial (becomes mfgSerialNumber)

We will fill out mfgSerialNumber in every case, even if it is the same as the shorter Serial. This means that the Access interface will now see an entry for every FullSerial field, although often it will be the same as the Serial field.

Confer with Paul explicitly about SSDS IDs 1429 and 1437, for which the serial numbers are in conflict. For example, 1429 numbers, SSDS first, are 37IM20000-1244 vs. 37IM21734-1244. If the latter is correct, no changes to the algorithm below are necessary.

The algorithm to follow for the merge (order matters) will be:

1. Present the mfgSerialNumber to Access as FullSerial.
2. If Instru has a full serial number, use it for mfgSerialNumber
    - If not, then if Instru has Serial, use that for mfgSerialNumber.

This algorithm works because (a) there are no cases where SSDS has a full serial number in mfgSerialNumber, but Instru only has a short Serial; and (b) there are no conflicts between Device and Instru (other than the two specifically mentioned above).

This algorithm can be implemented by assigning Serial whenever it exists, then assigning FullSerial whenever it exists. The former is done in SqlQueries#insertNewDevices and SqlQueries#updateExistingDevices; the latter is done in SqlQueriesFinal#updateToFullSerial.

Type

(See normalization strategy.)

The list of permitted types (so far, based on existing data as of 2/20) is as follows.

- Air Pressure Sensor
- Aircraft
- AUV
- Backscatterometer
- Backscatterometer/Fluorometer **(new)**
- Balloon
- Battery **(new)**
- BatteryLog

- Bioshutter
- Cable Monitor
- CO2 Monitor
- Combined
- Communication
- Communication/GPS **(new)**
- Compass
- Conductivity Sensor
- Controller
- Controller-CTD **(new)**
- Controller-Mooring Node
- Controller-Power
- CTD
- CTD Service
- Currents Sensor
- Currents Sensor-ADCP
- Data Acquisition Unit
- Data Acquisition Unit-SmartStar
- Drifter
- Drifter-Profiling
- Drifter-Surface
- Echo Sounder-Singlebeam **(new)**
- Echo Sounder-Multibeam **(new)**
- Electrical Sensor
- Electrical Sensor-Ground Fault
- Environment Monitor-System
- Flow Meter
- Flowthrough System
- Fluorometer
- Fluorometer/Nephalometer
- Glider
- GPS
- Humidity Sensor
- Hydrophone
- IMCTD
- Inductive Modem
- Inductive Modem-Surface
- Inductive Modem-Underwater
- Inductive Modem Cable Coupler **(new)**
- Kite
- Meteorology Package
- Metsys
- Model Service
- Mooring
- Navigation Service
- Navigation-AHRS
- Network Hub
- Network Hub-Power Controller
- Nitrogen Sensor
- Nitrogen Sensor-ISUS
- Optical
- Oxygen
- Oxygen Sensor
- Photometer-Bioluminescence
- platform
- Power Supply
- Power Supply-Battery
- Power Supply-Electric **(new)**
- Power Switch
- PowerSwitch
- Pressure Sensor
- pressureSensor
- Profiler
- Profiler Package

- Profiler-Moored
- Profiler-Ship
- Profiler-Suspended
- Propulsion Unit
- Pump **(new**)
- Radiometer
- Radiometer-Hyperspectral
- Radiometer-Longwave
- Radiometer-Multispectral
- Radiometer-Shortwave
- Release **(new**)
- ROV
- Satellite
- Scatterometer
- Scatterometer-LISST
- Ship
- Shutter-Antifouling
- Sound Speed Sensor
- Sonar-Sidescan Swath Bathymetry **(new)**
- Sonar-Scanning **(new)**
- Sonar-Subbottom **(new)**
- Spectrometer
- Temperature Sensor
- Test Device
- Test Entry-Instrument Type
- Test Simulated Instrument
- Toroid
- Towfish
- Transmissometer **(new)**
- Transponder **(new)**
- Velocity Sensor-DVL
- Wave Sensor
- Wave Sensor-Heave
- Wind Sensor

The following device types are also supported

- Test Entry-System
- Not Specific-Unknown
- Not Specified-TBD
- Not Specified-Undefined

Before beginning, confirm with Paul that the following changes are OK:

1. capitalizing first letters of each word
2. 'ADCP' to 'Current Sensors-ADCP'
3. 'modem' to 'Inductive Modem-Surface' or 'Inductive Modem-Underwater' as appropriate
4. OASIS 'Controller' (those with capital C) to 'Controller-Mooring Node'
5. separating 'backscatter' into the components below
    a. Scatterometer: Measures scattered emitted radiation (includes OBS, optical backscatterometer)
    b. Backscatterometer: measures reflected radiation from in-situ surfaces
6. 'fluor/turbidity' to 'Fluorometer/Nephalometer'
7. 'backscatter/fluorometer' to 'Backscatterometer/Fluorometer'
8. 'temperature/humidity' to 'Metereology Package'
9. 'Nitrate' to 'Nitrogen Sensor/ISUS'
10. 'Nitrate analyzer' to 'Nitrogen Sensor' (this is Satlantic unit)
11. 'Oxygen optode' (ignore case) to 'Oxygen Sensor'
12. 'Platform' to 'Toroid'
13. 'power source' to 'Power Supply-Electric'
14. 'pCO2' to 'CO2 Monitor'
15. 'transducer' to 'Transponder'
16. 'sounder' to 'Echo Sounder-Singlebeam'
17. 'shutter' to 'Shutter-Antifouling'
18. 'ICC' to 'Inductive Modem Cable Coupler'

19. 'pump' to 'Pump'
20. 'release' to 'Release'

> ⚠ The questions below were mostly unsettled as of press time.

Questions for Paul:

1. Are ECO BBSBs (161, 202) really Backscatterometers, or just Scatterometers?
    - If the former, change SSDS DeviceType for these to Backscatterometer
    - If the later, use existing SSDS DeviceType
2. Assumption: all HS2, VSFS units (he calls 'backscatter') are Backscatterometers
3. confirm HyperOCRs are Hyperspectral, i.e., controllable
    - If confirmed change 1346-1350 to be Hyperspectral
    - If not put these into appropriate category
4. Are any of our shutters not antifouling? Can we call them all Shutter-Antifouling?
5. How to change 'radiometer' to one of the following, as appropriate:
    a. Radiometer-Multispectral: measures a configurable series of wavelengths of radiation
    b. Radiometer-Hyperspectral: measures multiple specific wavelengths of radiation
    c. Radiometer-Shortwave: measures short-wave radiation ranges
    d. Radiometer-Longwave: measures long-wave radiation ranges

Add the types marked **(new)** above to the DeviceType table.

- Backscatterometer/Fluorometer: backscatterometer that additionally provides fluorescence data
- Battery: device that stores energy and makes it available in an electrical form; not mediated by electronics (see Power Supply-Battery)
- Communication/GPS: transmits signals from one place to another, and contains GPS locator
- Controller-CTD: controls acquisition of CTD data
- Echo Sounder-Singlebeam: acoustic system which transmits short acoustic pulses from single transmitter into the water column, and then detects echoes from impedance discontinuities (typically fish or plankton)
- Echo Sounder-Multibeam: acoustic system which transmits short acoustic pulses from multiple transmitters into the water column, and then detects echoes from impedance discontinuities (typically from the benthic layer)
- Inductive Modem Cable Coupler: connector for indcutive modems (ICC)
- Power Supply-Electric: power supply (delivers power to other equipment) that does not store energy
- Pump: mechanical device used to move liquids or gases
- Release: mechanism to disconnect two components, typically allowing a buoyant component in water to detach from an anchor and float to the surface
- Sonar-Sidescan Swath Bathymetry: fixed-mounted acoustic bathymetric sounding systems that sweep a wide path; also known as SSBS
- Sonar-Scanning: acoustic system that can move (examples are Articulating, Profiling, and Pencil Beam)
- Sonar-Subbottom: acoustic system designed to penetrate layers beneath the benthic surface (e.g., CHIRP)
- Transmissometer: device to measure the optical transparency of water (also turbidity sensor)
- Transponder: electrical device designed to receive a specific signal and automatically transmit a specific reply

Also, Change the name for device type 158 from 'Fluorometer/Nephalometer' to 'Fluorometer/Nephelometer', and change description to reflect new spelling.
Code to do all this is in SqlQueriesFinal#DeviceTypeInsert.

I have also created code to update the descriptions of the erroneously added deprecated terms at SqlQueriesFinal#UpdateDeprecated.

Implementation

The algorithm to follow for the merge (order matters) will be:

1. Change BOG name 'Controller' (match case) to 'Controller-Mooring Node'
2. SSDS names remain unchanged unless otherwise noted
   #+ If the Instru device has an SSDS entry, use the SSDS device type
    - If the Instru name matches (ignoring case) one in the above lists, use the list name.

These are coded in SqlQueriesFinal#DeviceTypeMerge.

1. Perform explicit corrections listed above.

We will copy the appropriate information over to our InterimTable (using the tempType column), from Instru or Device as appropriate, using the rules above.

Then we will do the entries requiring 'manual' correction (assuming Paul has OK'd the above), code for which is in SqlQueriesFinal#FixDeviceTypes:

At this point, all the tempType cells should be filled in.

Finally, the InterimTable will be used to overwrite the Device table.

Back to table...

Unchanged Fields

The following columns will not be changed in Phase 1, and so will be copied as is during the initial merging of the tables. This is done in SqlQueriesFinal#insertNewDevices and SqlQueriesFinal#updateExistingDevices.

Back to table...

Calibration organization

For phase I, maintain this material as it is now.

For phase II, consider making it compatible with manufacturer table. But realize that more information may need to be tracked for the calibration organization, and it may be a separate part of the same compnay as the manufacturer.

Back to table...

Features

Maintain this material without changes.

Back to table...

Pressure sensor

Maintain this material without changes.

Back to table...

Depth Rating

Maintain this material without changes.

Back to table...

Firmware/EPROM

Maintain this material without changes.

Back to table...

Memory

Maintain this material without changes.

Back to table...

Receive frequency

Maintain this material without changes.

Back to table...

Transmit frequency

Maintain this material without changes.

Back to table...

Enable code

Maintain this material without changes.

Back to table...

Release code

Maintain this material without changes.

Back to table...

Tilt option

Maintain this material without changes.

Back to table...

Purchased for

Maintain this material without changes.

Back to table...

Owner

Maintain this material without changes.

Back to table...

Custodian

Maintain this material without changes.

In the future, this field should become another Person field.

Back to table...

Date new

Maintain this material without changes.

Back to table...

PO

Maintain this material without changes.

Back to table...

Transaction

Maintain this material without changes (or delete this field).

Back to table...

Permanent comment

Maintain this material without changes (or delete this field).

Back to table...

manufacture web page

Maintain this material without changes.

In future, clarify its role and recode appropriately. (Is this the instrument web pages at the manufacturer's site?)

Back to table...

document dir

Maintain this material without changes.

SqlQueries

This page last changed on Mar 14, 2007 by graybeal.

> ⚠ These queries were tested against Northwind databases. For queries configured for SSDS_Metadata, see SqlQueriesFinal.

Finding Matching Instru Devices in SSDS (Information Only)

This finds full serial matches to SSDS mfgSerialNumber.

```
SELECT r2.MBARI_ID, d.id, r2.FullSerial, r2.Model, d.mfgModel, r2.Manufacturer, d.mfgName
FROM Device as d,
(
SELECT * FROM
(
SELECT     *
FROM         instru
WHERE     not exists (select null
FROM Device
WHERE Device.id = instru.[SSDS ID] )
) r1
WHERE EXISTS (select null
FROM Device
WHERE Device.mfgSerialNumber = r1.FullSerial)
) r2
WHERE d.mfgSerialNumber = FullSerial
ORDER BY r2.MBARI_ID
```

This finds short serial matches (lots of false positives).

```
SELECT r2.MBARI_ID, d.id, r2.Serial, r2.Model, d.mfgModel, r2.Manufacturer, d.mfgName
FROM Device as d,
(
SELECT * FROM
(
SELECT     *
FROM         instru
WHERE     not exists (select null
FROM Device
WHERE Device.id = instru.[SSDS ID] )
) r1
WHERE EXISTS (select null
FROM Device
WHERE Device.mfgSerialNumber = r1.Serial)
) r2
WHERE d.mfgSerialNumber = r2.Serial
ORDER BY r2.MBARI_ID
```

Match Instru to SSDS Devices

```
UPDATE Instru SET [SSDS ID] = 1359
WHERE MBARI_ID = 10

UPDATE Instru SET [SSDS ID] = 1567
WHERE MBARI_ID = 23

UPDATE Instru SET [SSDS ID] = 1568
WHERE MBARI_ID = 50

UPDATE Instru SET [SSDS ID] = 1222
WHERE MBARI_ID = 129
```

```
UPDATE Instru SET [SSDS ID] = 1273
WHERE MBARI_ID = 162

UPDATE Instru SET [SSDS ID] = 1337
WHERE MBARI_ID = 195

UPDATE Instru SET [SSDS ID] = 1279
WHERE MBARI_ID = 201

UPDATE Instru SET [SSDS ID] = 1356
WHERE MBARI_ID = 204

UPDATE Instru SET [SSDS ID] = 1470
WHERE MBARI_ID = 227

UPDATE Instru SET [SSDS ID] = 1562
WHERE MBARI_ID = 288

UPDATE Instru SET [SSDS ID] = 1382
WHERE MBARI_ID = 51

UPDATE Instru SET [SSDS ID] = 1406
WHERE MBARI_ID = 137

UPDATE Instru SET [SSDS ID] = 1267
WHERE MBARI_ID = 157

UPDATE Instru SET [SSDS ID] = 1456
WHERE MBARI_ID = 175

UPDATE Instru SET [SSDS ID] = 1457
WHERE MBARI_ID = 176

UPDATE Instru SET [SSDS ID] = 1314
WHERE MBARI_ID = 190

UPDATE Instru SET [SSDS ID] = 1459
WHERE MBARI_ID = 256
```

Change Technician in Instru to coenen

```
UPDATE instru
SET Technician = 'coenen'
WHERE Technician = 'pc'
```

Create Table of Legal Manufacturer Names

```
CREATE TABLE Manufacturers
(id int IDENTITY PRIMARY KEY,
label varchar (100) NOT NULL,
fullName varchar(100) NOT NULL,
oldNames varchar(250) NULL,
comments varchar(250) NULL)
GO

INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Aanderaa Data
Instruments','Aanderaa Data Instruments','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Axys Technologies','Axys
Technologies Inc.','','Axys Environmental Systems')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Biospherical
Instruments','Biospherical Instruments Inc.','','')
```

```
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Bluefin Robotics','Bluefin
Robotics Corporation','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Brook Ocean
Technology','Brook Ocean Technology Limited','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Crossbow
Technology','Crossbow Technology Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Garmin','Garmin','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('HOBI Labs','Hydro-Optics,
Biology, & Instrumentation Laboratories','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Magellan
Navigation','Magellan Navigation, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('MBARI','Monterey Bay
Aquarium Research Institute','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('McLane Research
Laboratories','McLane Research Laboratories, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('NOBSKA','NOBSKA','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('ORBCOMM','ORBCOMM,
Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values
('Paroscientific','Paroscientific, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('QUAKE Global','QUAKE
Global, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Satlantic','Satlantic
Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Sea-Bird
Electronics','Sea-Bird Electronics, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('SeaTech','SeaTech,
Inc.','','no longer exists')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Simrad','Simrad,
Inc','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Star Engineering for WHOI-
Asimet','Star Engineering','',
'Star Engineering is only authorized seller of ASIMETs')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('tbd','to be determined
value for manufacturer','','if totally unknown use unknown')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Teledyne
Benthos','Teledyne Benthos','Benthos','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Teledyne RD
Instruments','Teledyne RD Instruments','RD Instruments','' )
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('testing','test
entry','','use for test entries')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('UC Santa
Barbara','University of California, Santa Barbara','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('unknown','unknown
manufacturer','','use for permanently unknown values, else use tbd')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('WET Labs','WET
Labs','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('WHOI','Woods Hole
Oceanographic Institution','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Xantrex
Technology','Xantrex Technology Inc.','','')
```

Add New Device Table Entries from Instru

⚠ Double-check for Erich R's name in the Technician field before setting everything to 126.

```
/** Copy all Instru devices without SSDS IDs **/
INSERT INTO [Northwind].[graybeal].[Device]
([version], [uuid], [name], [description], [mfgName], [mfgModel], [mfgSerialNumber],
[infoUrlList], [PersonID_FK], [DeviceTypeID_FK], [Serial],
[Calibration_organization], [Features], [Pressure_sensor], [Depth_Rating],
[Firmware_EPROM], [Memory], [Receive_frequency], [Transmit_frequency],
[Enable_code], [Release_code], [Tilt_option], [Purchased_for],
```

```
[Owner], [Custodian], [Date_new], [PO], [Transaction_col],
[Permanent_comment], [document_dir], [osg_view], [MBARI_ID])

/** Notes for following:
Set Version, UUID to 0
Set name to null and description to Mfg + Model
* Leaving DeviceType to fix up later
* mfgSerialNumber gets Serial; overwrite with FullSerial (if exists) later
Transaction_col is set to null (always null anyway)
* Leaving MBARI_ID unchanged for now; later set it to id
**/
SELECT
0,'0',null,i.Manufacturer + ' ' + i.Model , i.Manufacturer, i.Model, i.Serial,
CAST (i.[manufacture web page]AS varchar(2048)), 126, 0, i.Serial,
i.[Calibration organization], i.Features, i.[Pressure sensor], i.[Depth Rating],
i.[Firmware/EPROM], i.Memory, i.[Receive frequency], i.[Transmit frequency],
i.[Enable code], i.[Release code], i.[Tilt option], i.[Purchased for],
i.Owner, i.Custodian, i.[Date new], i.PO, null,
i.[Permanent comment], i.[document dir], 1, i.MBARI_ID

FROM Instru as i
WHERE i.[SSDS ID] is null OR i.[SSDS ID] = 0

/** Copy SSDS ID for new devices back into the SSDS ID column of Instru. **/
UPDATE    i
SET              [SSDS ID] = d.id
FROM Device d, instru i
WHERE      d.MBARI_ID = i.MBARI_ID AND i.[SSDS ID] is null
```

Update Device Table Entries from Instru Devices

⚠ Double-check for Erich R's name in the Technician field before setting everything to 126.

```
/** Update all Instru devices with SSDS IDs **/
UPDATE    d
SET
/* Don't overwrite the model in Device if instru model is null */
d.mfgModel =
case when i.Model is not null then i.Model else d.mfgModel end,
d.mfgSerialNumber = i.Serial,
d.infoUrlList = CAST (i.[manufacture web page]AS varchar(2048)),
d.PersonID_FK = 126,
d.Serial = i.Serial,
d.Calibration_organization = i.[Calibration organization],
d.Features = i.Features,
d.Pressure_sensor = i.[Pressure sensor],
d.Depth_Rating = i.[Depth Rating],
d.Firmware_EPROM = i.[Firmware/EPROM],
d.Memory = i.Memory,
d.Receive_frequency = i.[Receive frequency],
d.Transmit_frequency = i.[Transmit frequency],
d.Enable_code = i.[Enable code],
d.Release_code = i.[Release code],
d.Tilt_option = i.[Tilt option],
d.Purchased_for = i.[Purchased for],
d.Owner = i.Owner,
d.Custodian = i.Custodian,
d.Date_new = i.[Date new],
d.PO = i.PO,
d.Transaction_col = null,
d.Permanent_comment = i.[Permanent comment],
d.document_dir = i.[document dir],
d.osg_view = 1,
```

```
d.MBARI_ID = i.MBARI_ID
FROM Device d, instru i
WHERE      d.MBARI_ID = i.MBARI_ID
```

Update mfgSerialNumber to Full Serial from Instru

```
/* Update mfgSerialNmber column with any FullSerial value */
UPDATE    d
SET              mfgSerialNumber = i.[FullSerial]
FROM Device d, instru i
WHERE      d.[MBARI_ID] = i.[MBARI_ID]
AND NOT (i.FullSerial is null OR i.FullSerial = '')
```

Create an Interim Working Table

```
/** Make Interim table **/
DROP TABLE TempDevFld
CREATE TABLE TempDevFld
(id int PRIMARY KEY,
tempMfg varchar (255) NULL,
tempType varchar (255) NULL
)
GO
/* Populate with all the device IDs */
INSERT INTO TempDevFld
(id)
SELECT d.id
FROM Device as d
```

Update Manufacturer (mfgName)

```
/** Update manufacturer names to ones in Manufacturers table **/
/* Grab names from Instru that match exactly */
UPDATE    t
SET       t.tempMfg = i.Manufacturer
FROM      TempDevFld t, instru i
WHERE      t.id = i.[SSDS ID] and i.Manufacturer in (SELECT label from Manufacturers) AND t.tempMfg
is null
/* Grab names from Device that match exactly */
UPDATE    t
SET       t.tempMfg = d.mfgName
FROM      TempDevFld t, Device d
WHERE      t.id = d.id and d.MfgName in (SELECT label from Manufacturers) AND t.tempMfg is null

/* Find names that are like the correct names */
/* First from instru */
UPDATE    t
SET       t.tempMfg = r1.label
FROM TempDevFld t,
(
SELECT t.id as tid, t.tempMfg, i.[SSDS ID] as iid, i.Manufacturer, m.label
FROM graybeal.TempDevFld t JOIN graybeal.instru i
ON (t.id = i.[SSDS ID]), graybeal.Manufacturers m
WHERE t.tempMfg is null AND
((substring(i.Manufacturer,1,3) = substring(m.label,1,3)
AND i.Manufacturer <> m.label
AND i.Manufacturer <> 'SeaTech' AND m.label <> 'SeaTech') OR
CHARINDEX(i.Manufacturer,m.label) > 1)
) r1
WHERE t.id = r1.tid AND t.tempMfg is null
```

```
/* Then from Device */
UPDATE    t
SET       t.tempMfg = r1.label
FROM TempDevFld t,
(
SELECT t.id as tid, t.tempMfg, d.id as did, d.mfgName, m.label
FROM graybeal.TempDevFld t JOIN graybeal.Device d
ON (t.id = d.id), graybeal.Manufacturers m
WHERE t.tempMfg is null AND
((substring(d.mfgName,1,3) = substring(m.label,1,3) AND (d.mfgName <> m.label)
AND d.mfgName <> 'SeaTech' AND m.label <> 'SeaTech') OR
CHARINDEX(d.mfgName,m.label) > 1)
) r1
WHERE t.id = r1.tid AND t.tempMfg is null



/* Do manual corrections */
/** Manual MfgName **/

UPDATE Device
SET mfgName = 'Aanderaa Instruments'
WHERE mfgName = 'Anderraa Instruments'

UPDATE Device
SET mfgName =
'Magellan Navigation'
WHERE mfgName LIKE
'Ashtech%'

UPDATE Device
SET mfgName =
'Star Engineering for WHOI-Asimet'
WHERE mfgName LIKE
'Asimet%'

UPDATE Device
SET mfgName =
'AXYS Technologies'
WHERE mfgName LIKE
'Triaxys%'

UPDATE Device
SET mfgName =
'UC Santa Barbara'
WHERE mfgName =
'UCSB'

UPDATE Device
SET mfgName =
'testing'
WHERE mfgName LIKE
'Fraben%'
```

For information only, this report generates a list of the original mfgName/Manufacturer against the newly created tempMfg.

```
/** Get relevant ID and manufacturer information from 3 tables **/
SELECT t.tempMfg, d.mfgName, i.Manufacturer, t.id, d.id, i.[SSDS ID], i.MBARI_ID
FROM
(
TempDevFld t
FULL OUTER JOIN Device d
ON t.id = d.id
)
```

```
FULL OUTER JOIN instru i
ON i.[SSDS ID] = t.id
ORDER BY tempMfg, mfgName
```

Update Model (mfgModel)

mfgModel column already contains BOG Model if any existed; this was done in initial merge of the two tables.

```
/* Update mfgModel to reflect manual corrections */
UPDATE Device
SET mfgModel =
'GPC16-HVS'
WHERE id = 1313

UPDATE Device
SET mfgModel =
'ECO FLNTUSB'
WHERE id = 1395

UPDATE Device
SET mfgModel =
null
WHERE mfgModel =
'(null)'

UPDATE Device
SET mfgModel =
replace (mfgModel, '-', ' ')
WHERE mfgModel LIKE 'ECO-FLNT%'

UPDATE Device
SET mfgModel =
'ECO Triplet'
WHERE mfgModel =
'triplet'

UPDATE Device
SET mfgModel =
'MMCv4'
WHERE mfgModel LIKE
'MMC v%'

UPDATE Device
SET mfgModel =
'Medusa Rev 2'
WHERE mfgModel LIKE
'Medusa R%'


UPDATE Device
SET mfgModel =
'Medusa Rev 2'
WHERE mfgModel LIKE
'Medusa v%'

UPDATE Device
SET mfgModel =
'OASIS Buoy'
WHERE mfgModel LIKE
'Fiberglass tor%'

UPDATE Device
SET mfgModel =
```

```
'E-meter'
WHERE mfgModel =
'Emeter'
```

Insert Additional Devices in Device Type Column

```
/* Insert new devices */
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Backscatterometer/Fluorometer', 'backscatterometer that additionally provides
fluorescence data')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Battery', 'device that stores energy and makes it available in an electrical form; not
mediated by electronics (see Power Supply-Battery)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Communication/GPS', 'transmits signals from one place to another, and contains GPS
locator')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Echo Sounder-Singlebeam', 'acoustic system which transmits short acoustic pulses from
single transmitter' +
' into the water column, and then detects echoes from impedance discontinuities (typically fish or
plankton)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Echo Sounder-Multibeam', 'acoustic system which transmits short acoustic pulses from
multiple transmitters' +
' into the water column, and then detects echoes from impedance discontinuities (typically from the
benthic layer)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Inductive Modem Cable Coupler', 'connector for indcutive modems (ICC)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Power Supply-Electric', 'power supply (delivers power to other equipment) that does not
store energy')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Pump', 'mechanical device used to move liquids or gases')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Release', 'mechanism to disconnect two components, typically allowing a buoyant
component in water'+
' to detach from an anchor and float to the surface')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Sidescan Swath Bathymetry', 'fixed-mounted acoustic bathymetric sounding systems
that sweep a wide path; also known as SSBS')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Scanning', 'acoustic system that can move (examples are Articulating, Profiling,
and Pencil Beam)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Subbottom', 'acoustic system designed to penetrate layers beneath the benthic
surface (e.g., CHIRP)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Transmissometer', 'device to measure the optical transparency of water (also turbidity
sensor)')
INSERT INTO graybeal.DeviceType(version ,Name, Description)
VALUES(0, 'Transponder', 'electrical device designed to receive a specific signal and automatically
transmit a specific reply')
```

Update Deprecated Device Descriptions

```
UPDATE DeviceType
SET description='Deprecated, please choose another type'
WHERE description is null
```

Update Nephelometer

```
/* Update the nephelometer spelling in DeviceType table */
update DeviceType
set name='Fluorometer/Nephelometer', description='fluorometer configured to also do nephelometry
(turbidity sensing)'
where id=158
```

Update Device Type Column by Merging

```
/* Update Device Type column */
/* Grab names from Device that exist (but do it through view) */
UPDATE    t
SET       t.tempType = dv.devType
FROM      TempDevFld t, devTypeView dv
WHERE      t.id = dv.id AND dv.devType is not null AND t.tempType is null
/* Grab names from Device that match exactly */

/* Change instru entries to be 'right' */
UPDATE i
SET i.Type = 'Controller-Mooring Node'
FROM Instru i
WHERE i.Type = 'Controller' AND (i.Model LIKE 'OASIS%' OR i.Model = 'MMC')

UPDATE i
SET i.Type = 'backscatterometer'
FROM instru i
WHERE Type = 'backscatter' AND (Model ='HS2' OR Model='VSFS')

UPDATE i
SET i.Type = 'scatterometer'
FROM instru i
WHERE Type = 'backscatter' AND Model ='ECO BBSB'

/* Find matching instru names */
/* Grab names from Instru that match exactly */
UPDATE    t
SET       t.tempType = i.Type
FROM      TempDevFld t, instru i
WHERE      t.id = i.[SSDS ID]
AND i.Type in (SELECT devType from devTypeView)
AND t.tempType is null
```

Manual Device Type Fixes

```
UPDATE    t
SET       t.tempType = 'Radiometer-Multispectral'
FROM      TempDevFld t
WHERE      t.id = 1381
UPDATE    t
SET       t.tempType = 'Inductive Model-Surface'
FROM      TempDevFld t
WHERE      t.id = 1319
UPDATE    t
SET       t.tempType = 'Controller-CTD'
FROM      TempDevFld t
WHERE      t.id = 1224
UPDATE    t
SET       t.tempType = 'Communication/GPS'
FROM      TempDevFld t
WHERE      t.id = 1323
```

```
UPDATE    t
SET       t.tempType = 'Current Sensors-ADCP'
FROM      TempDevFld t
WHERE     t.tempType = 'ADCP'

UPDATE t
set t.tempType = 'Fluorometer/Nephelometer'
FROM TempDevFld t
WHERE t.tempType = 'fluor/turbidity'
UPDATE t
set t.tempType = 'Backscatterometer/Fluorometer'
FROM TempDevFld t
WHERE t.tempType = 'backscatter/fluorometer'
UPDATE t
set t.tempType = 'Metereology Package'
FROM TempDevFld t
WHERE t.tempType = 'temperature/humidity'
UPDATE t
set t.tempType = 'Nitrogen Sensor/ISUS'
FROM TempDevFld t
WHERE t.tempType = 'Nitrate'
UPDATE t
set t.tempType = 'Nitrogen Sensor (this is Satlantic unit)'
FROM TempDevFld t
WHERE t.tempType = 'Nitrate analyzer'
UPDATE t
set t.tempType = 'Oxygen Sensor'
FROM TempDevFld t
WHERE t.tempType = 'Oxygen optode'
UPDATE t
set t.tempType = 'Toroid'
FROM TempDevFld t
WHERE t.tempType = 'Platform'
UPDATE t
set t.tempType = 'Power Supply-Electric'
FROM TempDevFld t
WHERE t.tempType = 'power source'
UPDATE t
set t.tempType = 'CO2 Monitor'
FROM TempDevFld t
WHERE t.tempType = 'pCO2'
UPDATE t
set t.tempType = 'Transponder'
FROM TempDevFld t
WHERE t.tempType = 'transducer'
UPDATE t
set t.tempType = 'Echo Sounder-Singlebeam'
FROM TempDevFld t
WHERE t.tempType = 'sounder'
UPDATE t
set t.tempType = 'Shutter-Antifouling'
FROM TempDevFld t
WHERE t.tempType = 'shutter'
UPDATE t
set t.tempType = 'Inductive Modem Cable Coupler'
FROM TempDevFld t
WHERE t.tempType = 'ICC'
UPDATE t
set t.tempType = 'Pump'
FROM TempDevFld t
WHERE t.tempType = 'pump'
UPDATE t
set t.tempType = 'Release'
FROM TempDevFld t
WHERE t.tempType = 'release'
set t.tempType = 'Backscatterometer'
```

```
FROM TempDevFld t
WHERE t.tempType = 'backscatter'
set t.tempType = 'Battery'
FROM TempDevFld t
WHERE t.tempType = 'battery'
set t.tempType = 'Transmissometer'
FROM TempDevFld t
WHERE t.tempType = 'transmissometer'
```

The final copy of device information back into the Device table (through the view, so it uses the FKs).

```
UPDATE dv
SET dv.devType = t.tempType
FROM devTypeView dv, TempDevFld t
WHERE dv.id = t.id
```

SqlQueriesFinal

This page last changed on Mar 14, 2007 by graybeal.

These queries have been tested against the SSDS_Metadata database. For queries configured for Northwind, see SqlQueries, but realize that those queries are somewhat older, and may work for John G but not others (because the tables were created for John G).

Finding Matching instruCopy Devices in Device (Information Only)

This finds full serial matches to Device mfgSerialNumber.

```
SELECT r2.MBARI_ID, d.id, r2.FullSerial, r2.Model, d.mfgModel, r2.Manufacturer, d.mfgName
FROM ssdsdba.Device as d,
(
SELECT * FROM
(
SELECT     *
FROM         instruCopy
WHERE     not exists (select null
FROM ssdsdba.Device
WHERE ssdsdba.Device.id = instruCopy.[SSDS ID] )
) r1
WHERE EXISTS (select null
FROM ssdsdba.Device
WHERE ssdsdba.Device.mfgSerialNumber = r1.FullSerial)
) r2
WHERE d.mfgSerialNumber = FullSerial
ORDER BY r2.MBARI_ID
```

This finds short serial matches (lots of false positives).

```
SELECT r2.MBARI_ID, d.id, r2.Serial, r2.Model, d.mfgModel, r2.Manufacturer, d.mfgName
FROM ssdsdba.Device as d,
(
SELECT * FROM
(
SELECT     *
FROM         instruCopy
WHERE     not exists (select null
FROM ssdsdba.Device
WHERE ssdsdba.Device.id = instruCopy.[SSDS ID] )
) r1
WHERE EXISTS (select null
FROM ssdsdba.Device
WHERE ssdsdba.Device.mfgSerialNumber = r1.Serial)
) r2
WHERE d.mfgSerialNumber = r2.Serial
ORDER BY r2.MBARI_ID
```

Match instruCopy to SSDS Devices

```
UPDATE instruCopy SET [SSDS ID] = 1359
WHERE MBARI_ID = 10

UPDATE instruCopy SET [SSDS ID] = 1567
WHERE MBARI_ID = 23

UPDATE instruCopy SET [SSDS ID] = 1568
```

```
WHERE MBARI_ID = 50

UPDATE instruCopy SET [SSDS ID] = 1222
WHERE MBARI_ID = 129

UPDATE instruCopy SET [SSDS ID] = 1273
WHERE MBARI_ID = 162

UPDATE instruCopy SET [SSDS ID] = 1337
WHERE MBARI_ID = 195

UPDATE instruCopy SET [SSDS ID] = 1279
WHERE MBARI_ID = 201

UPDATE instruCopy SET [SSDS ID] = 1356
WHERE MBARI_ID = 204

UPDATE instruCopy SET [SSDS ID] = 1470
WHERE MBARI_ID = 227

UPDATE instruCopy SET [SSDS ID] = 1562
WHERE MBARI_ID = 288

UPDATE instruCopy SET [SSDS ID] = 1382
WHERE MBARI_ID = 51

UPDATE instruCopy SET [SSDS ID] = 1406
WHERE MBARI_ID = 137

UPDATE instruCopy SET [SSDS ID] = 1267
WHERE MBARI_ID = 157

UPDATE instruCopy SET [SSDS ID] = 1456
WHERE MBARI_ID = 175

UPDATE instruCopy SET [SSDS ID] = 1457
WHERE MBARI_ID = 176

UPDATE instruCopy SET [SSDS ID] = 1314
WHERE MBARI_ID = 190

UPDATE instruCopy SET [SSDS ID] = 1459
WHERE MBARI_ID = 256
```

Insert Additional Devices in Device Type Column

Should be done before Instru merge. 

```
/* Insert new devices */
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Backscatterometer/Fluorometer', 'backscatterometer that additionally provides
fluorescence data')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Battery', 'device that stores energy and makes it available in an electrical form; not
mediated by electronics (see Power Supply-Battery)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Communication/GPS', 'transmits signals from one place to another, and contains GPS
locator')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Echo Sounder-Singlebeam', 'acoustic system which transmits short acoustic pulses from
single transmitter' +
' into the water column, and then detects echoes from impedance discontinuities (typically fish or
plankton)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
```

```
VALUES(0, 'Echo Sounder-Multibeam', 'acoustic system which transmits short acoustic pulses from
multiple transmitters' +
' into the water column, and then detects echoes from impedance discontinuities (typically from the
benthic layer)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Inductive Modem Cable Coupler', 'connector for indcutive modems (ICC)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Power Supply-Electric', 'power supply (delivers power to other equipment) that does not
store energy')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Pump', 'mechanical device used to move liquids or gases')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Release', 'mechanism to disconnect two components, typically allowing a buoyant
component in water'+
' to detach from an anchor and float to the surface')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Sidescan Swath Bathymetry', 'fixed-mounted acoustic bathymetric sounding systems
that sweep a wide path; also known as SSBS')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Scanning', 'acoustic system that can move (examples are Articulating, Profiling,
and Pencil Beam)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Sonar-Subbottom', 'acoustic system designed to penetrate layers beneath the benthic
surface (e.g., CHIRP)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Transmissometer', 'device to measure the optical transparency of water (also turbidity
sensor)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Transponder', 'electrical device designed to receive a specific signal and automatically
transmit a specific reply')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Test Entry-System', 'entry created for purpose of testing the system')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Not Specified-Unknown', 'an actual device for which the device type is not known (and
not knowable)')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Not Specified-TBD', 'entry for which the device type will be provided later')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Not Specified-Undefined', 'used for a device entry which does not yet define a specific
device')
INSERT INTO ssdsdba.DeviceType(version ,Name, Description)
VALUES(0, 'Controller-CTD', 'controls CTD data acquisition')
```

Update Deprecated Device Descriptions

These statements fix the newly added Device Descriptions (for the deprecated device types).

```
UPDATE ssdsdba.DeviceType
SET description='Deprecated, please choose another type'
WHERE description is null
```

Update Nephelometer

Fix a spelling mistake.

```
/* Update the nephelometer spelling in DeviceType table */
UPDATE ssdsdba.DeviceType
set name='Fluorometer/Nephelometer', description='fluorometer configured to also do nephelometry
(turbidity sensing)'
where name LIKE '%Fluor%Nep%'
```

Create Table of Legal Manufacturer Names

This isn't needed until later, but we're getting it out of the way now.

```
CREATE TABLE Manufacturers
(id int IDENTITY PRIMARY KEY,
label varchar (100) NOT NULL,
fullName varchar(100) NOT NULL,
oldNames varchar(250) NULL,
comments varchar(250) NULL)
GO

INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Aanderaa Data
Instruments','Aanderaa Data Instruments','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Axys Technologies','Axys
Technologies Inc.','Axys Environmental Systems','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Biospherical
Instruments','Biospherical Instruments Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Bluefin Robotics','Bluefin
Robotics Corporation','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Brook Ocean
Technology','Brook Ocean Technology Limited','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Crossbow
Technology','Crossbow Technology Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Garmin','Garmin','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('HOBI Labs','Hydro-Optics,
Biology, & Instrumentation Laboratories','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Magellan
Navigation','Magellan Navigation, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('MBARI','Monterey Bay
Aquarium Research Institute','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('McLane Research
Laboratories','McLane Research Laboratories, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('NOBSKA','NOBSKA','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Nortek AS','Nortek
AS','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('ORBCOMM','ORBCOMM,
Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values
('Paroscientific','Paroscientific, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('QUAKE Global','QUAKE
Global, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Satlantic','Satlantic
Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Sea-Bird
Electronics','Sea-Bird Electronics, Inc.','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('SeaTech','SeaTech,
Inc.','','no longer exists')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Simrad','Simrad,
Inc','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Star Engineering for WHOI-
Asimet','Star Engineering','',
'Star Engineering is only authorized seller of ASIMETs')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('tbd','to be determined
value for manufacturer','','if totally unknown use unknown')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Teledyne
Benthos','Teledyne Benthos','Benthos','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Teledyne RD
Instruments','Teledyne RD Instruments','RD Instruments | RDI',''   )
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('testing','test
entry','','use for test entries')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('UC Santa
Barbara','University of California, Santa Barbara','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('unknown','unknown
manufacturer','','use for permanently unknown values, else use tbd')
```

```
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('WET Labs','WET
Labs','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('WHOI','Woods Hole
Oceanographic Institution','','')
INSERT INTO Manufacturers (label, fullName, oldNames, comments) values ('Xantrex
Technology','Xantrex Technology Inc.','','')
```

Add New Device Table Entries from instruCopy

This script creates new Device entries for the instru devices not already in SSDS.

> ⚠ Double-check for Erich R's name in the Technician field before setting everything to 126. Kevin: Paul says he sees Erich's name in the Technician field, but I just copied the latest instru from BOG and I don't see it. Can you resolve this apparent conflict? – jbg 3/14 6 PM.

Note this is updated now to set the device type to an actual entry.
This fails if the DeviceType table has not been updated.

```
/** Copy all instruCopy devices without SSDS IDs **/
INSERT INTO [SSDS_Metadata].[ssdsdba].[Device]
([version], [uuid], [name], [description], [mfgName], [mfgModel], [mfgSerialNumber],
[infoUrlList], [PersonID_FK], [DeviceTypeID_FK], [Serial],
[Calibration_organization], [Features], [Pressure_sensor], [Depth_Rating],
[Firmware_EPROM], [Memory], [Receive_frequency], [Transmit_frequency],
[Enable_code], [Release_code], [Tilt_option], [Purchased_for],
[Owner], [Custodian], [Date_new], [PO], [Transaction_col],
[Permanent_comment], [document_dir], [osg_view], [MBARI_ID])

/** Notes for following:
Set Version, UUID to 0
Set name to null and description to Mfg + Model
* Leaving DeviceType as the TBD entry, to fix up later
* mfgSerialNumber gets Serial; overwrite with FullSerial (if exists) later in scripts
Transaction_col is set to null (always null anyway)
* Leaving MBARI_ID unchanged for now; later can set it to id
**/
SELECT
0,NEWID(),null,i.Manufacturer + ' ' + i.Model , i.Manufacturer, i.Model, i.Serial,
CAST (i.[manufacture web page]AS varchar(2048)), 126, (select id from ssdsdba.DeviceType where name
LIKE '%TBD%'), i.Serial,
i.[Calibration organization], i.Features, i.[Pressure sensor], i.[Depth Rating],
i.[Firmware/EPROM], i.Memory, i.[Receive frequency], i.[Transmit frequency],
i.[Enable code], i.[Release code], i.[Tilt option], i.[Purchased for],
i.Owner, i.Custodian, i.[Date new], i.PO, null,
i.[Permanent comment], i.[document dir], 1, i.MBARI_ID

FROM instruCopy as i
WHERE i.[SSDS ID] is null OR i.[SSDS ID] = 0

/* Copy SSDS ID for new devices back into the SSDS ID column of instruCopy. */
UPDATE    i
SET              [SSDS ID] = d.id
FROM ssdsdba.Device d, instruCopy i
WHERE      d.MBARI_ID = i.MBARI_ID AND (i.[SSDS ID] is null OR i.[SSDS ID] = 0)
```

UPDATE ssdsdba.Device Table Entries from instruCopy Devices

This updates the Device table for the instru entries that we know are SSDS entries.

```
/** Update all instruCopy devices that have SSDS IDs **/
UPDATE    d
```

```
SET
/* Don't overwrite the model in Device if instruCopy model is null */
d.mfgModel =
case when i.Model is not null then i.Model else d.mfgModel end,
d.mfgSerialNumber = i.Serial,
d.infoUrlList = CAST (i.[manufacture web page]AS varchar(2048)),
d.PersonID_FK = 126,
d.Serial = i.Serial,
d.Calibration_organization = i.[Calibration organization],
d.Features = i.Features,
d.Pressure_sensor = i.[Pressure sensor],
d.Depth_Rating = i.[Depth Rating],
d.Firmware_EPROM = i.[Firmware/EPROM],
d.Memory = i.Memory,
d.Receive_frequency = i.[Receive frequency],
d.Transmit_frequency = i.[Transmit frequency],
d.Enable_code = i.[Enable code],
d.Release_code = i.[Release code],
d.Tilt_option = i.[Tilt option],
d.Purchased_for = i.[Purchased for],
d.Owner = i.Owner,
d.Custodian = i.Custodian,
d.Date_new = i.[Date new],
d.PO = i.PO,
d.Transaction_col = null,
d.Permanent_comment = i.[Permanent comment],
d.document_dir = i.[document dir],
d.osg_view = 1,
d.MBARI_ID = i.MBARI_ID
FROM ssdsdba.Device d, instruCopy i
/* If osg_view = 1 then this device was created in the steps above, no need to recopy. */
WHERE      d.id = i.[SSDS ID] AND d.osg_view = 0
```

Update mfgSerialNumber to Full Serial from instruCopy

The short serial was copied in the merge step above, if it existed. This overwrites with FullSerial from instru, if it exists.
This version corrects a bug from the Northwind version, where it only tested for MBARI_IDs matching.

```
/* Update mfgSerialNmber column with any FullSerial value */
UPDATE    d
SET              mfgSerialNumber = i.[FullSerial]
FROM ssdsdba.Device d, instruCopy i
WHERE      d.id = i.[SSDS ID]
AND NOT (i.FullSerial is null OR i.FullSerial = '')
```

Create an Interim Working Table

This makes the TempDevFld table, which is used to cache completed mfgName and deviceType data.

```
/** Make Interim table **/
DROP TABLE TempDevFld
CREATE TABLE TempDevFld
(id int PRIMARY KEY,
tempMfg varchar (255) NULL,
tempType varchar (255) NULL
)
GO
/* Populate with all the device IDs */
INSERT INTO TempDevFld
(id)
```

```
SELECT d.id
FROM ssdsdba.Device as d
```

Update Manufacturer (mfgName)

Makes the Manufacturer names consistent.

```
/** Update manufacturer names to ones in Manufacturers table **/
/* Grab names from instruCopy that match exactly */
UPDATE    t
SET       t.tempMfg = i.Manufacturer
FROM      TempDevFld t, instruCopy i
WHERE      t.id = i.[SSDS ID] and i.Manufacturer in (SELECT label from Manufacturers) AND t.tempMfg
is null
/* Grab names from Device that match exactly */
UPDATE    t
SET       t.tempMfg = d.mfgName
FROM      TempDevFld t, ssdsdba.Device d
WHERE      t.id = d.id and d.MfgName in (SELECT label from Manufacturers) AND t.tempMfg is null

/* Find names that are like the correct names */
/* First from instruCopy */
UPDATE    t
SET       t.tempMfg = r1.label
FROM TempDevFld t,
(
SELECT t.id as tid, t.tempMfg, i.[SSDS ID] as iid, i.Manufacturer, m.label
FROM TempDevFld t JOIN instruCopy i
ON (t.id = i.[SSDS ID]), Manufacturers m
WHERE t.tempMfg is null AND
((substring(i.Manufacturer,1,3) = substring(m.label,1,3)
AND i.Manufacturer <> m.label
AND i.Manufacturer <> 'SeaTech' AND m.label <> 'SeaTech') OR
CHARINDEX(i.Manufacturer,m.label) > 1)
) r1
WHERE t.id = r1.tid AND t.tempMfg is null

/* Then from Device */
UPDATE    t
SET       t.tempMfg = r1.label
FROM TempDevFld t,
(
SELECT t.id as tid, t.tempMfg, d.id as did, d.mfgName, m.label
FROM TempDevFld t JOIN ssdsdba.Device d
ON (t.id = d.id), Manufacturers m
WHERE t.tempMfg is null AND
((substring(d.mfgName,1,3) = substring(m.label,1,3) AND (d.mfgName <> m.label)
AND d.mfgName <> 'SeaTech' AND m.label <> 'SeaTech') OR
CHARINDEX(d.mfgName,m.label) > 1)
) r1
WHERE t.id = r1.tid AND t.tempMfg is null

/* Copy the results back to Device table */
UPDATE d
SET d.MfgName = t.tempMfg
FROM ssdsdba.Device d, TempDevFld t
WHERE d.id = t.id AND t.tempMfg is not null

/* Do manual corrections */
/** Manual MfgName -- we now update in the Device table directly **/

/* This appears to have no effect, item likely corrected in solstice Device table */
UPDATE ssdsdba.Device
SET mfgName = 'Aanderaa Instruments'
```

```
WHERE mfgName = 'Anderraa Instruments'

UPDATE ssdsdba.Device
SET mfgName =
'Magellan Navigation'
WHERE mfgName LIKE
'Ashtech%'

UPDATE ssdsdba.Device
SET mfgName =
'Axys Technologies'
WHERE mfgName LIKE
'AXYS Technologies'

UPDATE ssdsdba.Device
SET mfgName =
'Star Engineering for WHOI-Asimet'
WHERE mfgName LIKE
'Asimet%'

/* Pick up some ASIMETs that are wrong in Device */
UPDATE ssdsdba.Device
SET mfgName =
'Star Engineering for WHOI-Asimet'
WHERE id = 1250 OR id = 1251

UPDATE ssdsdba.Device
SET mfgName =
'AXYS Technologies'
WHERE mfgName LIKE
'Triaxys%'

UPDATE ssdsdba.Device
SET mfgName =
'Teledyne RD Instruments'
WHERE mfgName =
'RDI'

UPDATE ssdsdba.Device
SET mfgName =
'UC Santa Barbara'
WHERE mfgName =
'UCSB'

UPDATE ssdsdba.Device
SET mfgName =
'testing'
WHERE mfgName LIKE
'Fraben%'
```

For information only, this report generates a list of the original mfgName/Manufacturer against the newly created tempMfg. It would have to be run before copying the data back to the Device table above, and would not include the final adjustments shown below that.

```
/** Get relevant ID and manufacturer information from 3 tables **/
SELECT t.tempMfg, d.mfgName, i.Manufacturer, t.id, d.id, i.[SSDS ID], i.MBARI_ID
FROM
(
TempDevFld t
FULL OUTER JOIN ssdsdba.Device d
ON t.id = d.id
)
FULL OUTER JOIN instruCopy i
ON i.[SSDS ID] = t.id
ORDER BY tempMfg, mfgName
```

Update Model (mfgModel)

mfgModel column already contains BOG Model if any existed; this was done in initial merge of the two tables.

```
/* Update mfgModel to reflect manual corrections */
UPDATE ssdsdba.Device
SET mfgModel =
'GPC16-HVS'
WHERE id = 1313

UPDATE ssdsdba.Device
SET mfgModel =
'ECO FLNTUSB'
WHERE id = 1395

UPDATE ssdsdba.Device
SET mfgModel =
null
WHERE mfgModel =
'(null)'

UPDATE ssdsdba.Device
SET mfgModel =
replace (mfgModel, '-', ' ')
WHERE mfgModel LIKE 'ECO-FLNT%'

UPDATE ssdsdba.Device
SET mfgModel =
'ECO Triplet'
WHERE mfgModel =
'triplet'

UPDATE ssdsdba.Device
SET mfgModel =
'MMCv4'
WHERE mfgModel LIKE
'MMC v%'

UPDATE ssdsdba.Device
SET mfgModel =
'Medusa Rev 2'
WHERE mfgModel LIKE
'Medusa R%'


UPDATE ssdsdba.Device
SET mfgModel =
'Medusa Rev 2'
WHERE mfgModel LIKE
'Medusa v%'

UPDATE ssdsdba.Device
SET mfgModel =
'OASIS Buoy'
WHERE mfgModel LIKE
'Fiberglass tor%'

UPDATE ssdsdba.Device
SET mfgModel =
'E-meter'
WHERE mfgModel =
'Emeter'
```

UPDATE temporary device type Column by Merging

This works on the temporary table TempDevFld.

```
/* UPDATE temporary Device Type column */
/* Grab names from Device that already exist (but do it through view) */
UPDATE    t
SET       t.tempType = dv.devType
FROM      TempDevFld t, devTypeView dv
WHERE      t.id = dv.id AND dv.devType is not null AND t.tempType is null

/* Change instruCopy entries to be 'right' */
UPDATE i
SET i.Type = 'Controller-Mooring Node'
FROM instruCopy i
WHERE i.Type = 'Controller' AND (i.Model LIKE 'OASIS%' OR i.Model = 'MMC')

UPDATE i
SET i.Type = 'backscatterometer'
FROM instruCopy i
WHERE Type = 'backscatter' AND (Model ='HS2' OR Model='VSFS')

UPDATE i
SET i.Type = 'scatterometer'
FROM instruCopy i
WHERE Type = 'backscatter' AND Model ='ECO BBSB'

UPDATE i
SET i.Type = 'Inductive Modem-Surface'
FROM instruCopy i
WHERE Type = 'modem' AND Model ='SIM'

UPDATE i
SET i.Type = 'Fluorometer/Nephelometer'
FROM instruCopy i
WHERE Type = 'fluor/turbidity'

UPDATE i
SET i.Type = 'Backscatterometer/Fluorometer'
FROM instruCopy i
WHERE Type = 'backscatter/fluorometer'

UPDATE i
SET i.Type = 'Metereology Package'
FROM instruCopy i
WHERE Type = 'temperature/humidity'

UPDATE i
SET i.Type = 'Nitrogen Sensor/ISUS'
FROM instruCopy i
WHERE Type = 'Nitrate'

UPDATE i
SET i.Type = 'Nitrogen Sensor (this is Satlantic unit)'
FROM instruCopy i
WHERE Type = 'Nitrate analyzer'

UPDATE i
SET i.Type = 'Oxygen Sensor'
FROM instruCopy i
WHERE Type = 'Oxygen optode'

UPDATE i
SET i.Type = 'Toroid'
FROM instruCopy i
```

```
WHERE Type = 'Platform'

UPDATE i
SET i.Type = 'Power Supply-Electric'
FROM instruCopy i
WHERE Type = 'power source'

UPDATE i
SET i.Type = 'CO2 Monitor'
FROM instruCopy i
WHERE Type = 'pCO2'

UPDATE i
SET i.Type = 'Transponder'
FROM instruCopy i
WHERE Type = 'transducer'

UPDATE i
SET i.Type = 'Echo Sounder-Singlebeam'
FROM instruCopy i
WHERE Type = 'sounder'

UPDATE i
SET i.Type = 'Shutter-Antifouling'
FROM instruCopy i
WHERE Type = 'shutter'

UPDATE i
SET i.Type = 'Inductive Modem Cable Coupler'
FROM instruCopy i
WHERE Type = 'ICC'

UPDATE i
SET i.Type = 'Pump'
FROM instruCopy i
WHERE Type = 'pump'

UPDATE i
SET i.Type = 'Release'
FROM instruCopy i
WHERE Type = 'release'

UPDATE i
SET i.Type = 'Backscatterometer'
FROM instruCopy i
WHERE Type = 'backscatter' OR Type = 'backscatterometer'

UPDATE i
SET i.Type = 'Battery'
FROM instruCopy i
WHERE Type = 'battery'

UPDATE i
SET i.Type = 'Transmissometer'
FROM instruCopy i
WHERE Type = 'transmissometer'


/* Find matching instruCopy names */
/* Grab names from instruCopy that match exactly */
UPDATE    t
SET       t.tempType = i.Type
FROM      TempDevFld t, instruCopy i
WHERE      t.id = i.[SSDS ID]
AND i.Type in (SELECT devType from devTypeView)
AND (t.tempType is null OR t.tempType = 'Not Specified-TBD')
```

Manual Device Type Fixes

This updaets all the errant device types (still in the temporary table).

```
UPDATE    t
SET       t.tempType = 'Radiometer-Multispectral'
FROM      TempDevFld t
WHERE      t.id = 1381
UPDATE    t
SET       t.tempType = 'Inductive Modem-Surface'
FROM      TempDevFld t
WHERE      t.id = 1319
UPDATE    t
SET       t.tempType = 'Controller-CTD'
FROM      TempDevFld t
WHERE      t.id = 1224
UPDATE    t
SET       t.tempType = 'Communication/GPS'
FROM      TempDevFld t
WHERE      t.id = 1323
UPDATE    t
SET       t.tempType = 'Currents Sensor-ADCP'
FROM      TempDevFld t
WHERE     t.tempType = 'ADCP'
```

The final copy of device information back into the Device table (through the view, so it uses the FKs).

```
UPDATE d2
SET d2.DeviceTypeID_FK = r1.typeid
FROM ssdsdba.Device d2,
(
select t.id as id, d.id as typeid
from TempDevFld t left outer join ssdsdba.DeviceType d
on t.tempType = d.name
) r1
WHERE d2.id = r1.id
```

For information only, here is code to look at the final device types to see if they are sensible.

```
/* Final check to see if it's all working */

select d.id, dtv.id, dtv.devType, d.name, d.mfgName, d.mfgModel, d.description
from
devTypeView as dtv JOIN ssdsdba.Device as d
on dtv.id = d.id
order by devType, d.id
```

Design Review Notes

This page last changed on Feb 12, 2007 by graybeal.

Review Meeting November 21, 2006

Agenda

1. Project Background (5 minutes - 2:05)
2. 4 Use Case Reviews (5 minutes - 2:10)
3. By Use Case Discuss Activity Diagram and Proposed Solution
    a. Create New Device in Instru (10 minutes - 2:20)
    b. Edit Device in Instru (10 minutes - 2:30)
    c. Create New Device in SSDS (15 minutes - 2:45)
    d. Edit Device in SSDS (15 minutes - 3:00)

Attendees:

- Kevin Gomes
- Reiko Michisaki
- Neil Conner
- Francisco Chavez
- Rich Schramm
- John Graybeal
- Aaron Marburg

Meeting Notes:

Project Background

In our initial effort, we will be merging data from SSDS to BOG and back, where both databases have overlapping data.
Eventually we'll try to merge back some of Paul's unique information back into the SSDS database.

Will the information in SSDS be easy to get to? Yes, we'll have pages. (Desire expressed to change those.)

Clarification of databases: "Instru" and "Trans" database in BOG is in fact Paul's database. (The MS Access database no longer exists, it has been moved into BOG although the Access interface is still used.)

This is a two-way system, but with the catch that not all the SSDS instrument entries will go into BOG. Paul will have right of refusal of an instrument into BOG (which is a subset).

Checkpoints will exist so that Paul can approve changes, before they are included. We were trying to be non-intrusive on Paul's current processes and data store. Someday it would be nice to have a single table in one database, but for now we aren't going to try to fit all the instruments into the operational (BOG) database. (SSDS will contain all the devices.)

One could implement a single table with a flag to limit Paul's view? No, because some of Paul's fields do not map gracefully to SSDS tables, e.g., the people fields. Could you put everything from SSDS into BOG, but limit what Paul sees? Yes, but it's a little intrusive, we wanted to have minimal impact on OSG in the first stages.

Have we talked to accounting? They track all fixed assets, we should consider that at the same time (e.g., add a field to manage data of interest to Jim R).

What's the allowed latency? Let's look at this in context of a use case.

Does Paul have write permission to BOG? Just to those two tables.

Where are calibration files tracked? In BOG Transaction table there are some calibration references. SSDS has the Resource field, but we are not using it yet. Keep calibration files in mind as a key capability for the end product. We would use a URL to point to the calibration file(s) when we do implement it.

Note it would be nice to have a physical store for all the calibration files, so they are all in a consistent location.

Put list of variables in each table on the Wiki.

Noted that table in BOG is not complete, not all items have values entered.

Will we be able to access this device information via an interface in SSDS? Yes. Why was it originally in BOG? Just to be in a SQL database.

Would be nice for anyone to have access to BOG database (not currently the case). There are individuals who are interested in access. Probably leave the model with individual authenticated access.

There is everyone/guest access to the SSDS SQL interface, and the web interface is wide open.

4 Use Case Reviews: By Use Case Discussion

Create New Device in Instru

This is the shortest latency case, where a new instrument in BOG should immediately check the contents of SSDS for congruent information.

When Paul creates a new device he'll enter a new transaction. At this time we're not trying to map the transaction data, we'll revisit that later.

Does the BOG application provide all the fields needed to search for a match, so that syntactical differences are normalized? We'll go through the list as a first step to normalize the current entries, working with Paul to maximize consistency. In the device creation page for SSDS it will be constrained, unlike the open field currently used.

What about fields in SSDS like instrument type that BOG doesn't care about but needs (to match keys)? There are only 5 used for search, BOG has them although some normalization between the two is required. We'll agree on rules with Paul and program the synchronization accordingly.

What is the server in SSDS, a SQL server? Can these talk? Yes, they're on the same server.

Edit Device in Instru

What is latency? Probably within an hour is fine. (We don't know of any 'side-by-side' use cases, where two people might be working simultaneously in the two tables.)

Does SSDS provide a last updated field? No, it doesn't. We'll be turning on history tables that would enable access to that functionality. This is a very common question Paul asks.

Create New Device in SSDS

How quickly does a new device in SSDS have to be accessible in BOG? We think not quickly.
Confirm this. (See also previous section for similar question.)

Once an 'accept with edit' happens, another email will get issued once the edit happens? Yes.

AI-> If device is rejected and later edited, it will exist in SSDS but not BOG, so the update action would get confused.
Response: Yes, many actions may get confused if editing happens before OSG acts. Comment added (may require redesign so data does not go into table until BOG response received.

AI-> Note that the "insert new instrument in instru BOG" step will trigger the other use case, so if we use triggers we'll have to trap that case.
Response: "Insert new instrument in Instru BOG" is a trigger of both use cases, so the other use case is not triggered by this action.

We will create an access-controlled capability to create devices (not wide open like it is now).

How will trigger accomplish sending mail? A stored procedure can do it. We don't implement SQLMail. Wouldn't be too hard to implement in current version, but will require investigation, and then would change for 2005 implementation (Neil will be trying to implement around January).

Edit Device in SSDS

AI-> "No" Arrow to final box comes from wrong box, should come from diamond.
Response: Could not determine specifically what this refers to, no similar problem was found. Other improvements may have taken care of it.

AI-> "Is change approved?" should be explicit about who approves change.
Response: Done.

What happens if OSG accidentally rejects a change, instead of accepting it? We can manually fire the stored procedure.

If the SSDSAdmin does not react to the notification of device edit, what happens? This introduces a latency issue, if no action is taken it introduces an unexplained discrepancy.
AI-> Look at this more closely.
Response: This is correct. It has been noted on the diagram.

Will a global update cause many triggers? Umm, yes.

Final Discussion

Francisco wants to answer:

- Where is everything?
- What is the complete history of any given thing?

Batch processing would be extremely valuable addition. (Currently is not working, John says Paul says.)

Each one of these use cases fires companion use case, so mechanism needed to trap those (and the bulk update case).

Do you want this specifically for one SQL server, or do you want it deployable anywhere? (Raises trigger/ procedures architectural question.)

Recommendation: Keep as little work going on as possible inside the trigger. If one database is down, trigger will fail and this will be reflected in UI. If you don't need atomic real-time transaction, this will create headaches. Trigger will wait for reply from stored procedures. Dirty bit might be a better answer, with a second table to capture the row/timestamp for a later SQL or Perl job. Making it external this way gives you a lot more control over processing and less likelihood of hosing Paul's user interface. Stored procedures couples DBs too tightly.

Watch error paths on flow charts – how will notification occur on failures? (This is potentialy complex on a multiple-database system.)

If you wanted to go to one table someday, look into creating a view right now that does it. Can you even do an Access-linked table to a view? We think so. We think it is tractable to put things together into one table, but it is more invasive to Paul's practices. Would make the job a lot easier to be working within a single database. Even if we need to change it out later, it will be easy to copy it over when it is needed.

Response: Based on the discussion, we have investigated and are currently pursuing a single-database solution. (We have successfully tested the use of Access through views.)

Requirements

This page last changed on Nov 29, 2006 by graybeal.

Requirements

1. Introduction
   This document outlines the requirements for the Asset Configuration and Tracking Consolidation task, 2006 proposal with charge number 900626.
2. General Description
   A general description of the task is found in the Project Proposal. The tasks as described in that proposal are:

   > The first step in solving this issue is to locate and document the different systems that are currently handling asset tracking and information. These systems include: SSDS, Paul Coenen's Database, BOG, and LOBO. After identifying the different systems, the information that is contained in those projects must also be detailed. From these details duplicated information can be identified.
   > The next step is to then identify the different applications that are utilized to edit and maintain this information. The functionality of these interfaces will be documented to make sure that the coordinated solution meets all the needs of the individual applications and their users.
   > With these pieces in hand, a decision can be made on how to best handle the coordinated system and a design will be created for this purpose. The solution will be reviewed and then implemented.
   > **Deliverables** The final deliverable should be an application (or small set of applications) that can handle the asset tracking and configuration information so all users will go to one location to find out information about various instruments that MBARI maintains.

   As currently envisioned based on information learned to date (2006.11.05), the initial effort will not incorporate LOBO, and Paul Coenen's database is considered to be the same as the BOG 'instru' database.
3. Functional Requirements
   a. Address Existing Problems
      i. Minimize errors in SSDS database
         New and changed instrument data should be verified and accurate.
      ii. Avoid duplicated entries in SSDS database
         The same instrument should not appear twice in the SSDS database.
      iii. Simplify data entry tasks
         Make user data entry operations as simple as possible (minimizes mistakes, makes user happy).
      iv. Minimize data entry tasks
         Data entered in one interface should populate the other wherever feasible.
   b. General Goals
      i. Reflect Instru Changes To SSDS Database
         Changes to the Instru database should be reflected in the SSDS database.
      ii. Reflect SSDS Changes To Instru Database
         Changes to the SSDS database should be reflected in the Instru database, to the extent they are of interest to users of the Instru database.
   c. Use Cases
      The following requirements can be mapped to corresponding use cases on the Use Cases page.
      i. OSG Operator Creates New Device in Instru Database
         An OSG operator will use the Instru datbase application (MS Access) to enter a new device in the Instru data table in BOG database.
      ii. OSG Operator Edits A Device in Instru Database
         An OSG operator will use the Instru datbase application (MS Access) to edit existing device information in the Instru data table in BOG database.
      iii. SSDS Operator Creates New Device in SSDS Database
         An SSDS (or other) operator will enter a new device in the SSDS device table in the SSDS database.
      iv. SSDS Operator Edits A Device in SSDS Database
         An SSDS (or other) operator will edit existing device information in the Device data table in the SSDS database.
4. Interface Requirements

    a. The existing MS Access interface to Paul Coenen's Instru database must be supported, or its equivalent functions provided to the satisfaction of OSG.
    b. The existing SSDS services must remain available.
    c. Planned improvements to the SSDS device entry service will be implemented (e.g., authentication, validation).
    d. Operation of the existing SSDS device search interface should be clarified.
5. Performance Requirements
    a. The changes must support the expected number of devices to be entered over the next 5 years, without significantly impacting usability.
6. Design Constraints
    a. Paul Coenen must maintain authority to control changes to the instruments of interest to him (i.e., those currently in the Instru database).
    b. Multiple users, from OSG, Development, and Engineering will be responsible for creating and editing device metadata. The system must gracefully minimize data entry errors, and allow review processes (automated and manual) to catch errors when they do occur.
7. Other non-functional attributes
    a. Security
        i. The author of each change to the SSDS database should be identifiable.
        ii. Changes to device metadata should be reviewable by a designated authority for that device record.
    b. Binary Compatibility
    c. Reliability
    d. Maintainability
    e. Portability
    f. Extensibility
        i. To the extent possible, the system should take into account the likely addition of other instrument collections in the future.
    g. Reusability
    h. Application Affinity/Compatibility
    i. Resource Utilization
    j. Serviceability

Comments

I'd like to see a capability for managing calibartion information that people who process data from instruments currently may keep in properties files. Cases where this is done include data for the radiometers and fluorometers.

Posted by mccann at Dec 05, 2006.

This sounds like a great functionality. Any reason it can't be done within the current SSDS data model using the "Resource" field?

Posted by amarburg at Dec 07, 2006.

Use Cases

This page last changed on Nov 06, 2006 by kgomes.

Use Case Name	Create New Device in Instru
Related Requirements	OSG Operator can create a new device in the Instrument database (BOG)
Goal In Context	An OSG operator needs to be able to enter a new device in the insrument Microsoft Access application
Preconditions	Device does not already exist in the data store
Successful End Condition	New device entry is inserted and information shows up in both Access app and SSDS
Failed End Condition	The device does not show up in either application
Primary Actors	OSG Operator
Secondary Actors	
Trigger	New Device will be put in use
Included Cases	
Base Use Cases	
Main Flow	1. OSG Operator opens Access Application 2. OSG Operator clicks on button to insert new device 3. OSG Operator fills out fields to describe new device 4. OSG Operator clicks button to create new device 5. A notification is sent to the OSG Operator and SSDS Administrator of device creation
Extensions	1. After OSG Operator clicks on button to create new device, validation fails and creation is rejected

Use Case Name	Edit Device in Instru
Related Requirements	OSG Operator can update device information in Instrument Database (BOG)
Goal In Context	The OSG Operators need to be able to change the information about a device in the Instrument Database Access application
Preconditions	Device needs to exist in the data store where the Access application reads from
Successful End Condition	Device information is updated in the data store and changes show up in SSDS and Access app
Failed End Condition	No device information was updated
Primary Actors	OSG Operator
Secondary Actors	
Trigger	
Included Cases	
Base Use Cases	OSG Operator finds incorrect or out of date device information
Main Flow	1. OSG Operator opens Access Application 2. OSG Operator searches for device 3. OSG Operator changes form fields 4. OSG Operator clicks button to udpate device 5. Device information is updated in the data store 6. Notification of change is sent to SSDS Admin describing device information update
Extensions	1. After OSG Operator clicks on update button, information validation fails and no update takes place

Use Case Name	Create New Device in SSDS
Related Requirements	SSDS Operator shall be able to create a new device in SSDS.
Goal In Context	The system needs to have the capability for users (with appropriate permissions) add new devices to the data store.
Preconditions	1. Device not already stored in the data store 2. SSDS Operator has appropriate permissions to create new device
Successful End Condition	New device is entered in the data store
Failed End Condition	No new device is entered in the data store
Primary Actors	SSDS Operator
Secondary Actors	OSG Operator
Trigger	A new Device needs to be put into service.
Included Cases	1. Authenticate User 2. Authorize User
Base Use Cases	
Main Flow	1. SSDS Operator logs into SSDS web application 2. SSDS Operator selects link to create new device 3. include::Authenticate User 4. include::Authorize User 5. SSDS Operator fills out the appropriate information 6. SSDS Operator clicks on Submit to attempt to create new device 7. Fields are validated 8. Information entered is checked against data store: • If no UUID was entered, create one automatically • Make sure serial number is not null • Make sure manufacturer name, model and serial number do not already exist 9. New device is entered in data store. 10. SSDS Operator, SSDS Admin and OSG Operator are notified of new device creation. 11. In notification to OSG Operator, he/she is presentd option of importing to Instrument Database 12. OSG Operator clicks on link to approve transfer to Access application 13. Data is transferred to Access data store
Extensions	1. If field validations fail, user is presented with another try 2. If information entered violates rule, user is presented with another try

Use Case Name	Edit Device in SSDS
Related Requirements	SSDS Operator or Software Agent shall have interface to edit device information in SSDS.
Goal In Context	Both human and software processes need interfaces to update device information in SSDS.
Preconditions	Device already exist in data store
Successful End Condition	Device information is updated in the data store
Failed End Condition	No device information is updated in the data store
Primary Actors	• SSDS Operator • Software Agent
Secondary Actors	
Trigger	Information about a device has changed or is incorrect

Included Cases	1. Authenticate User 2. Authorize User
Base Use Cases	
Main Flow	1. a. SSDS Operator logs into SSDS web application b. SSDS Operator searches for existing device c. SSDS Operator selects link to edit device d. include::Authenticate User e. include::Authorize User f. SSDS Operator changes the appropriate information g. SSDS Operator clicks on Submit to attempt to update device h. Fields are validated i. Information entered is checked against data store: • Make sure UUID is valid if changed. • If manufacturer information is changed, make sure it does not match up with another device already in the system. j. Device information is changed in the data store. k. SSDS Operator and SSDS Admin are notified of change in device information. l. If device present in Instrument Database store, OSG Operator is notified of changes in SSDS and has option of approving or editing SSDS changes, if editing takes place, OSG Operator becomes SSDS Operator and starts Main Flow again. If changes are approved, edits are propogated to the Instrument Database. 2. a. Software Agent sends calls Device update service with device information. b. Fields are validated c. Information entered is checked against data store: • Make sure UUID is valid if changed. • If manufacturer information is changed, make sure it does not match up with another device already in the system. d. Device information is changed in the data store. e. SSDS Operator and SSDS Admin are notified of change in device information. f. If device present in Instrument Database store, OSG Operator is notified of changes in SSDS and has option of approving or editing SSDS changes, if editing takes place, OSG Operator becomes SSDS Operator and starts Main Flow again. If changes are approved, edits are propogated to the Instrument Database.
Extensions	1. a. If field validations fail, user is presented with another try b. If information entered violates rules, user is presented with another try 2. a. If field validations fail, service throws exception back to Software Agent

	b. If information entered violates rules, service throws exception back to Software Agent.

Use Case Name	Authenticate User
Related Requirements	
Goal In Context	
Preconditions	
Successful End Condition	
Failed End Condition	
Primary Actors	
Secondary Actors	
Trigger	
Included Cases	
Base Use Cases	
Main Flow	
Extensions	

Use Case Name	Authorize User
Related Requirements	
Goal In Context	
Preconditions	
Successful End Condition	
Failed End Condition	
Primary Actors	
Secondary Actors	
Trigger	
Included Cases	
Base Use Cases	
Main Flow	
Extensions	