THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN

Tailoring a Component Framework With User-Defined Connectors

OMG Component Information Day

March 18, 2013

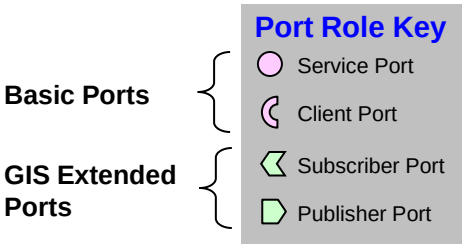
Mark Hayman
Consulting Systems Architect
Northrop Grumman Corporation

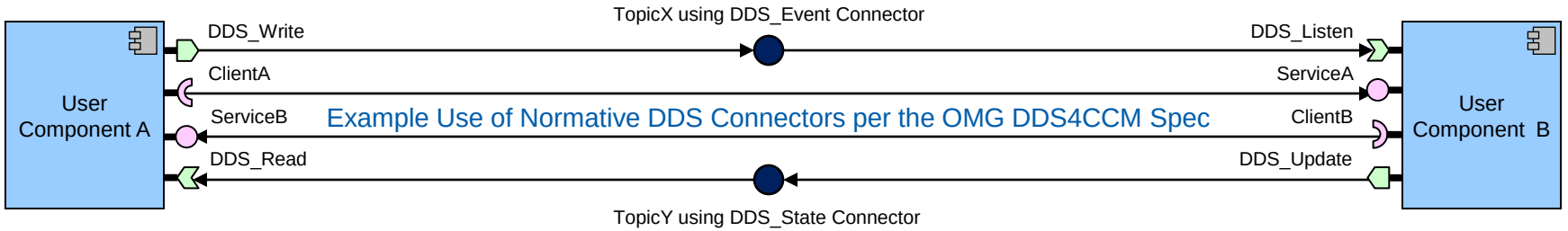
Component Framework Customization

Leveraging Generic Interaction Support (GIS) for Specific Domains

NORTHROP GRUMMAN

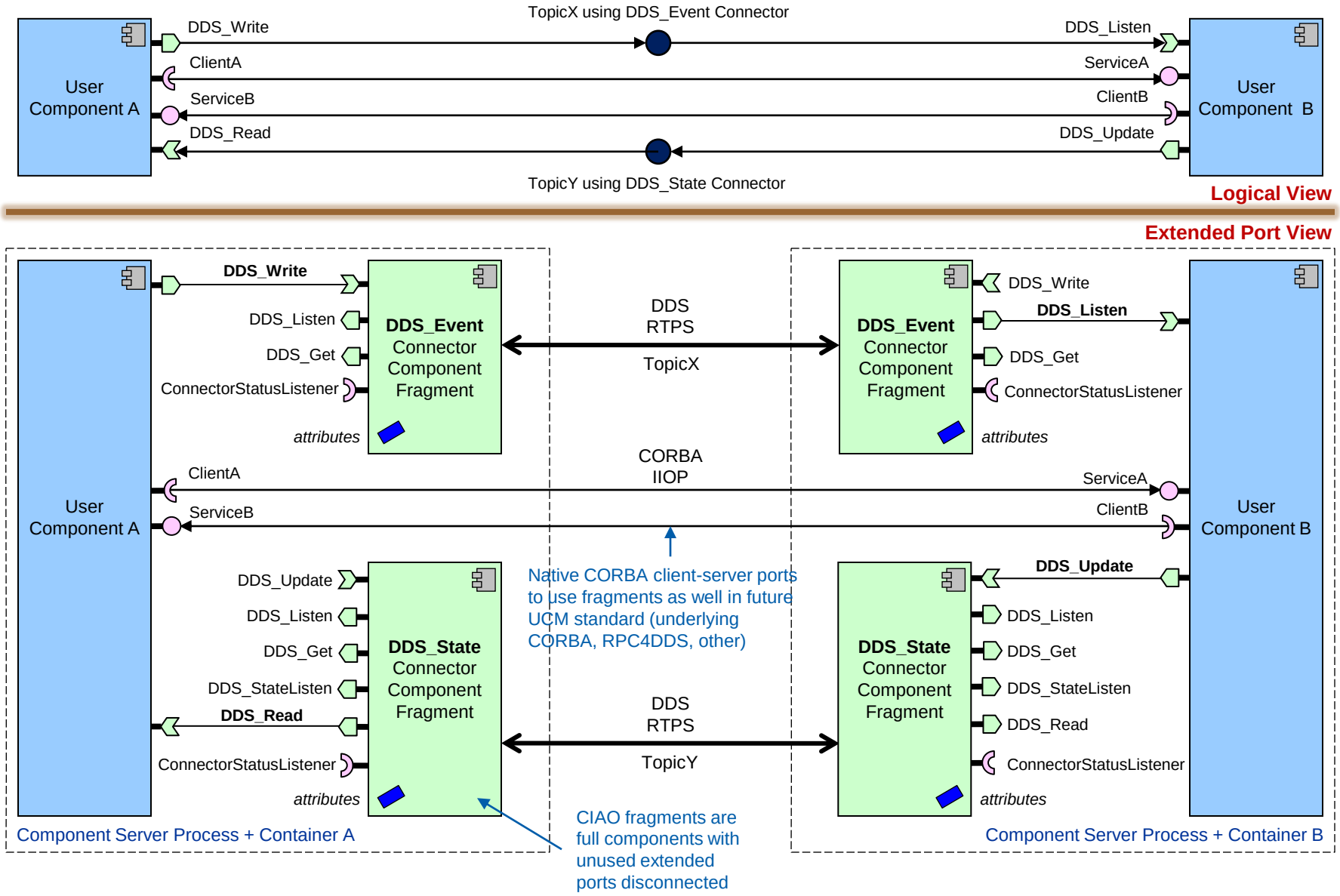
- GIS enables the encapsulation of domain specific functionality, drivers, alternative middleware, communication interfaces, or most anything within a “connector”
  - Connectors are concretely implemented as component “fragments” that are “collocated” with application components in the same component server process + container at deployment
- Connectors offer two key interfaces
  - **Portability Interface**: API exposed vertically to application components, defined in IDL as one or more aggregated sets of “local” interfaces called an “extended port,” and compiled to simple virtual function calls (very low overhead)
  - **Interoperability Interface**: Underlying, hidden, horizontal communication protocol between all like fragment instances, implemented however desired w/o impact to application components
- Custom connectors can be designed for unique applications
  - By leveraging a GIS compliant LwCCM component framework (or future UCM) that allows you to build your own connectors
  - By leveraging model driven architecture (MDA) tools that make design, IDL generation, and deployment of components that use instantiations of them very easy





DDS4CCM Connectors

Logical Extended Ports Implemented by Underlying Fragments



Ref: DDS_Event and DDS_State Fragments

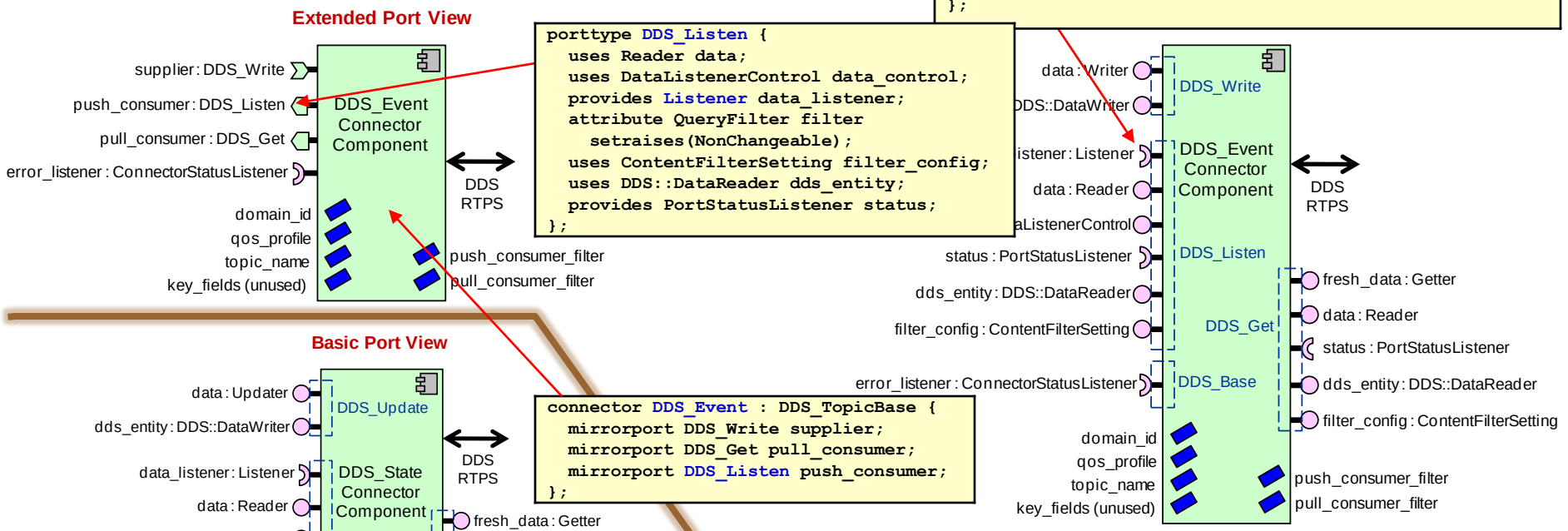
Graphical Representation From ccm_dds.idl

NORTHROP GRUMMAN

```
local interface Listener {
  void on_one_data (in T datum, in ReadInfo info);
  void on_many_data (in TSeq data, in ReadInfoSeq infos);
};
```

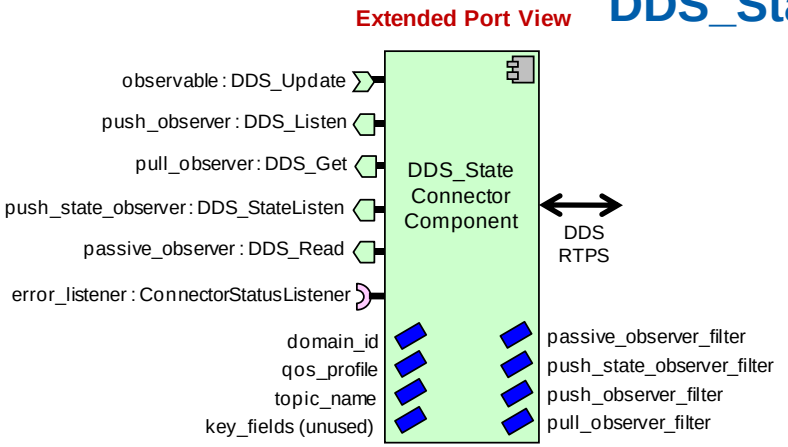
```
porttype DDS_Listen {
  uses Reader data;
  uses DataListenerControl data_control;
  provides Listener data_listener;
  attribute QueryFilter filter
    setraises(NonChangeable);
  uses ContentFilterSetting filter_config;
  uses DDS::DataReader dds_entity;
  provides PortStatusListener status;
};
```

```
connector DDS_Event : DDS_TopicBase {
  mirrorport DDS_Write supplier;
  mirrorport DDS_Get pull_consumer;
  mirrorport DDS_Listen push_consumer;
};
```



DDS_Event

DDS_State



So, How Do You Build Your Own Connector?

NORTHROP GRUMMAN

- Define your connector in IDL
  - Define types, exceptions and local interfaces (API) plus an IDL templated module
  - Then extended port types (aggregations of local interfaces) & your connector(s)
- Create a model of the connector in your chosen [1]CBDDS modeling tool
  - Currently, UML profiles for CBDDS are offered by Zeligsoft CX for CBDDS (PrismTech) and the Artisan Studio IDL Profile (Atego)
  - Both tools offer IDL generation from a model for most of the IDL 3.5 spec, so much of the prior IDL definition can be done *in* a model, as well as D&C CDP/CDD descriptor gen
  - Add your connector model to your tool installation for access by app developers
    - Installed as a model library for Zeligsoft, and an importable package for Artisan
- Build the source implementation of your connector
  - We use the user extensible GIS connector framework in the open source CIAO implementation of LwCCM, DDS4CCM & AMI4CCM from the DOC Group
  - Result is a template based implementation that application developers can instantiate with a user-defined type (message, interface, etc.)

[1]Component Based DDS (CBDDS) defines an integrated suite of 7 OMG open standards:
LwCCM, DDS, DDS4CCM, AMI4CCM, CORBA, IDL and D&C

Custom GIS Connector Examples

Built and In Service at Northrop Grumman

NORTHROP GRUMMAN

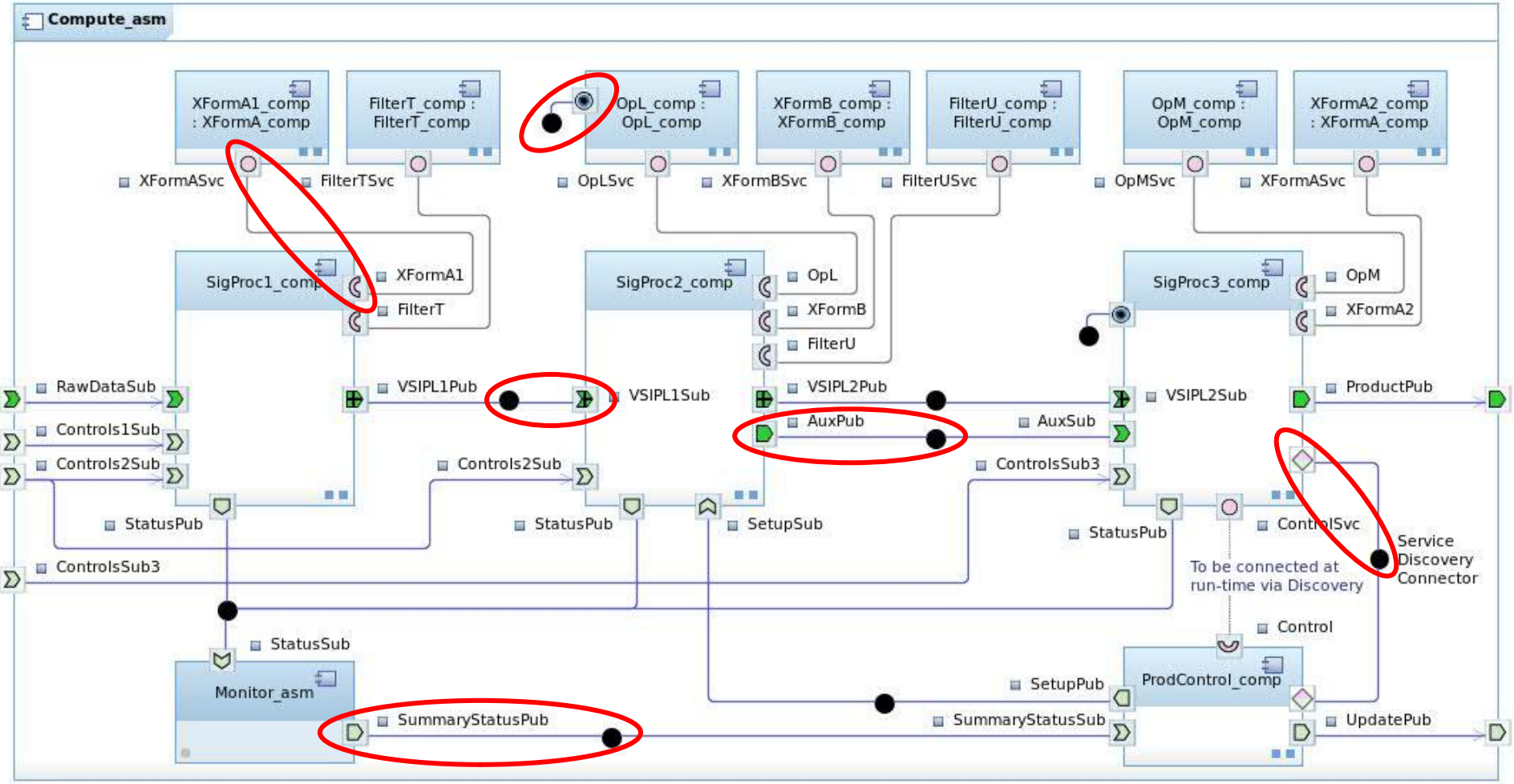
- Publish Subscribe Attachment Transfer (PSAT) connector
  - High performance, general purpose & location independent pub-sub transport of wideband data with DDS signaling
- Signal Processing Data Model (SPDM) connector
  - PSAT extension to support transport of OMG [1]VSIPL++ or VSIPL blocks and views for signal and image processing applications
- Application Instrumentation (AI) connector
  - CBDDS PSM simplification of the DDS PSM, providing a very easy to use encapsulation of in-development OMG AI standard for binary data instrumentation
- Discovery connector
  - Directory services access to support dynamic, run-time registration, discovery/lookup and binding of component service endpoints and topic data

[1]VSIPL = Vector, Signal & Image Processing Library

Component Assembly Diagram

Example Showing Basic and Extended Port Types & Connectors

NORTHROP GRUMMAN



Basic Port Types

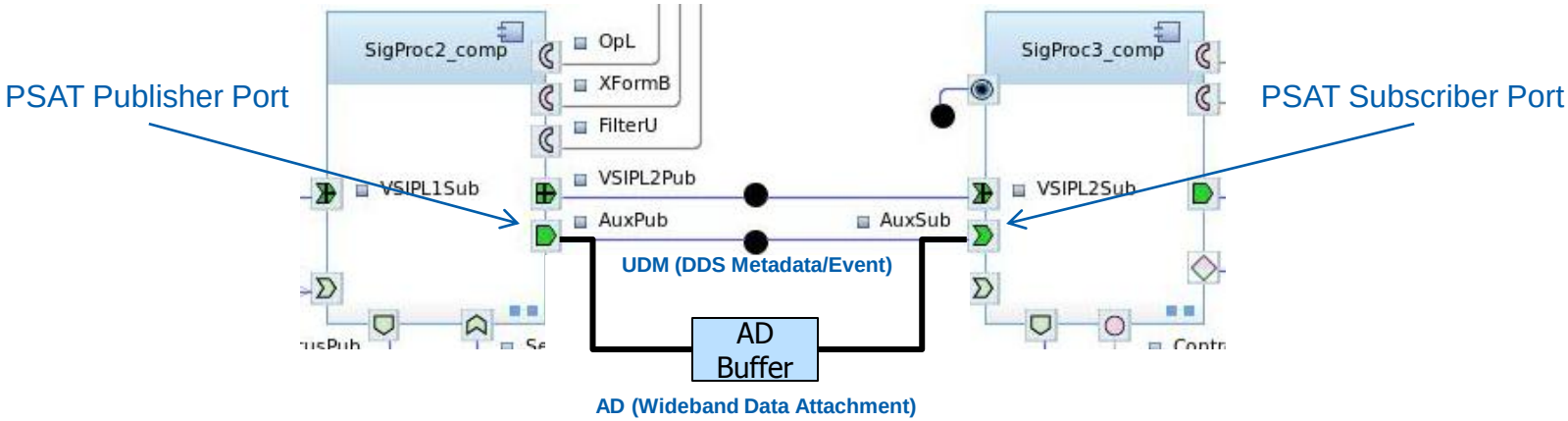
- Service (Facet)
- Client (Receptacle) Sync or Async (AMI4CCM)

Extended Port Types

- DDS_Write, DDS_Update
- DDS_Listen, DDS_Read, DDS_StateListen, DDS_Get
- PSAT_Write
- PSAT_Listen
- SPDM PSAT_Base::PSAT_Write
- SPDM PSAT_Base::PSAT_Listen
- AI_Save
- Discover (Data or Services)

PSAT Connector

- DDS4CCM DDS extension to support the efficient and high performance transport of large, wideband data for HPC (high performance computing) applications
- PSAT = Publish Subscribe Attachment Transfer
  - Name borrows from the email paradigm of sending an opaque, intact file attached to an email that describes what the attached file is
    - Large/wideband Attached Data (AD) buffer ~= attached file
    - User Defined Message (UDM) metadata message ~= email text body
  - UDM metadata & IPC event sent via DDS, AD buffer sent separately over a wideband fabric such as shared memory or Infiniband
  - UDM explicitly described in IDL, AD implicitly described by standard “core” UDM message header parameters that must be included in every PSAT message per connector design
- Design extends the DDS4CCM standard CIAO open source DDS_Event connector implementation with a new PSAT_Event connector
  - A few additional API calls beyond the standard DDS4CCM APIs



Why PSAT and Not Standard DDS?

NORTHROP GRUMMAN

- DDS messages must be specified by a single IDL struct or union definition
  - No means to separately specify metadata vs. data, send them independently, and then associate them on the receiving side for simultaneous single-callback delivery
- HPC applications such as signal & image processing need low latency, finer grained control over memory layout and copying for publishers & subscribers
  - The DDS standard APIs abstract away sample buffer management & location for ease of use
  - Consecutive sent/received sample buffers have no guaranteed relationship to one another in memory, and are typically copied many times between writer & reader
    - RTI Connext Messaging DDS: Three times on same host, four times between hosts
- No ability to send DDS sub-samples in order to specify contiguous sample buffer layout or support many-to-one aggregation into a common target sample buffer
  - No exposed DDS API capability for enhanced sub-sample functionality or scatter/gather
- NGC and RTI did a multi-month “large data” design study and concluded that DDS API extensions would be required to meet HPC requirements
  - Primarily for zero/single copy AD buffer lifecycle management, which adds both unnecessary complication to DDS, and more potential for abuse of concurrent memory access controls
- Some DDS products work over high speed fabrics like Infiniband, but they are still IP-based (IPoIB)

PSAT Wideband AD Buffer Transport Options

NORTHROP GRUMMAN

- Configurable transport options include:
  - Zero-copy Shared Memory – preferred intra-node option, highest throughput
  - Single-Copy Shared Memory – optional lower performance intra-node option, supporting the ability to mirror possible OFED RDMA data transformations during transport
  - OFED (OpenFabrics Enterprise Distribution) compliant wideband fabric
    - OFED is an RDMA (Remote Direct Memory Access) API standard for Linux/Windows (PSAT can sustain 3.1 GBytes/sec over QDR Infiniband)
    - Available OFED fabrics include Infiniband, Serial Rapid IO or 10/40 GbE
    - Scatter/gather and other in-transport data transformations are possible
  - TCP/IP - development/debug only, for VM SDK development when fabric unavailable
  - File System - future
  - NOTE: Connector APIs seen by business logic classes are the same for all
    - Defined as standard GIS local IDL interfaces – full location-independent deployment
- PSAT connector configuration parameters set by two means:
  - Some are PSAT connector attributes, settable during the CBDDS deployment planning phase from an MDA tool (and generated to a D&C deployment plan)
  - Most configuration options *currently* specified in a PSAT config file
    - Allows necessary per-connector-fragment settings, which DAnCE supports
    - Per generic GIS connector concept, all fragments must share common connector settings (although a “per port property override” feature is pending in the MDA tools)

SPDM Connector

CBDDS Support for the OMG VSIPL/VSIPL++ Standards

NORTHROP GRUMMAN

- SPDM = Signal Processing Data Model
- A DDS4CCM connector type built via a minor extension to the PSAT connector
  - Uses IDL 3.5 syntax for connector inheritance
  - Adds an additional connector attribute to PSAT, and additional standard UDM header info
- Extends the general purpose transport layer PSAT connector
  - VSIPL/VSIPL++ knowledgeable, supporting location-independent “View” data model transport
  - PSAT remains application layer and data model agnostic – purely a transport layer connector
- Direct support for a high level VSIPL/VSIPL++ data model struct in an extended PSAT UDM message
  - PSAT_Header, an additional SPDM_Header, plus user-defined fields
- Adds a custom optional PSAT reader interceptor to handle complexities involved in transferring VSIPL/VSIPL++ blocks & views from one component to another
  - All data transport is between location independent components, transparent to the application layer and APIs
    - Same OS process, different OS processes on same compute node, different compute nodes – all deployment planning time decisions w/no code impact
    - Enables coarse-grained HPEC multi-node parallelism at component level, supplementing fine-grained intra-node parallelism covered by VSIPL and VSIPL++

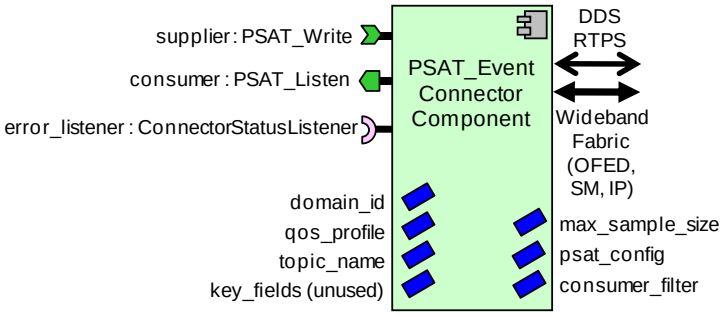
Ref: PSAT_Event and SPDM_Event Fragments

Graphical Representation From ccm_psat.idl & ccm_spdm.idl

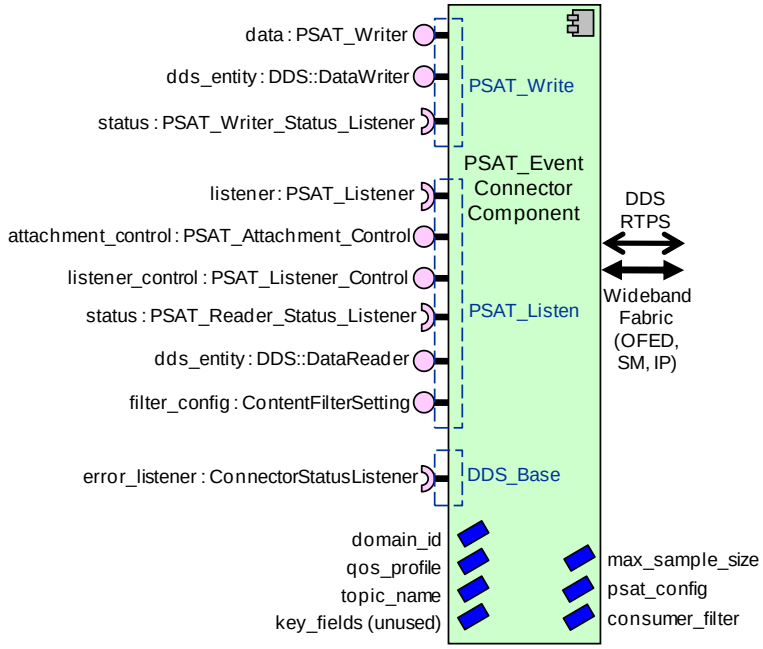
NORTHROP GRUMMAN

PSAT_Event

Extended Port View

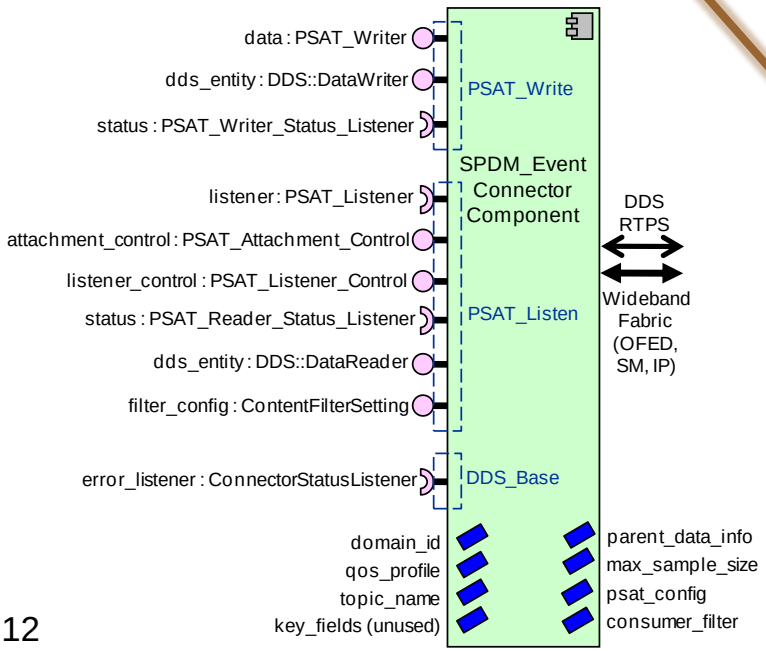


Basic Port View

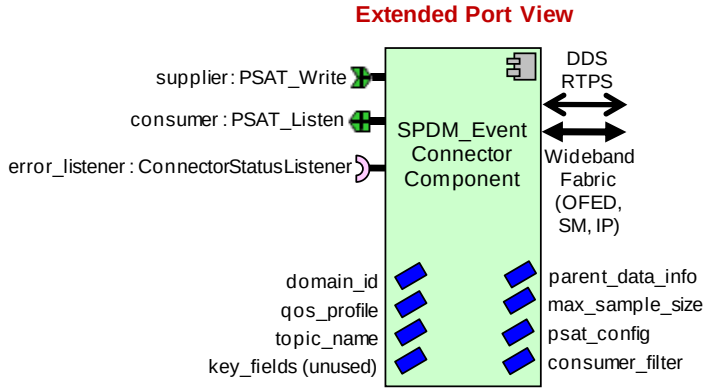


SPDM_Event

Basic Port View



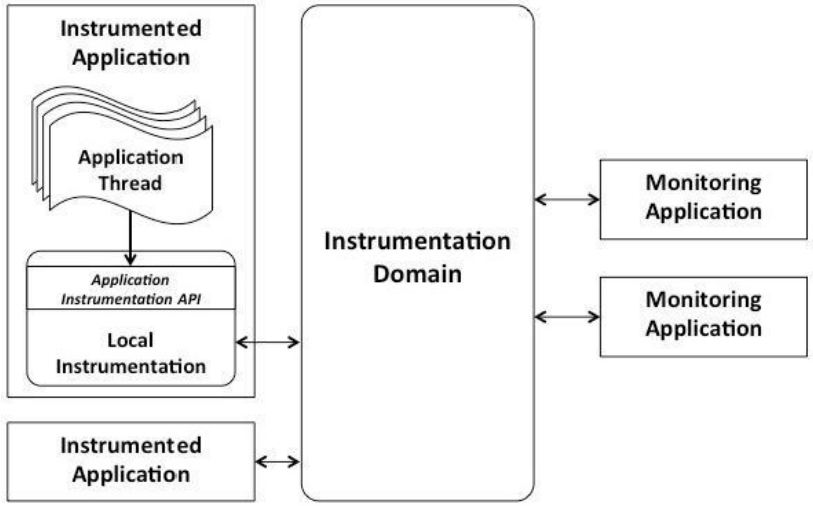
Extended Port View



Application Instrumentation (AI) Connector

NORTHROP GRUMMAN

- Provides a very easy to use capability to add run-time DRE binary data instrumentation to components
  - Supplements distributed text-based logging typically used for DRE debug
- Application Instrumentation (AI) is an in-development OMG draft standard
  - Under the C4I DTF
- The AI connector is basically a CBDDS PSM simplification of the DDS PSM in the draft spec



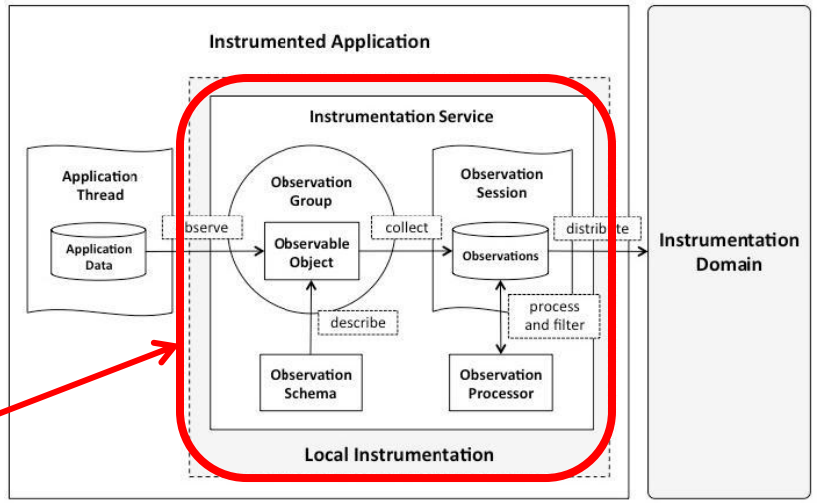
DDS PIM to PSM Mapping

Ref: OMG Document c4i/2013-02-01

8.2.1.1.2 Instrumentation entities

Instrumentation entities defined by the API are uniquely associated with entities in the DDS middleware.

API entity	Middleware entity
InstrumentationService	*DDS_DomainParticipant*
ObservationSession	Publishing thread
ObservableSchema	*DDS_Topic*
ObservationGroup	*DDS_Publisher*
ObservableObject	Instance on *DDS_Topic*



- The AI connector, in conjunction with the LwCCM container and DDS4CCM connector framework, encapsulates & implements Instrumentation Service functionality

AI Connector Definition and Use

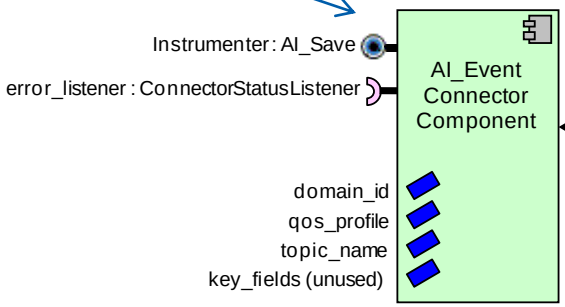
NORTHROP GRUMMAN

- **AI Connector Definition:**

```
// AI_Event – some IDL details removed for clarity

local interface AI_Control {
  CCM_AI::ReturnCode_t bind(
    in long long objStruct);
  CCM_AI::ReturnCode_t observe();
  CCM_AI::ReturnCode_t unbind();
};

porttype AI_Save {
  uses Typed::AI_Control data;
  uses DDS::DataWriter dds_entity;
};

connector AI_Event : CCM_DDS::DDS_TopicBase {
  mirrorport Typed::AI_Save instrumenter;
};
```

Portability Interface: Single port (see left)

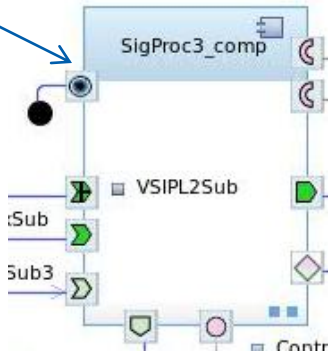
Extended Port View



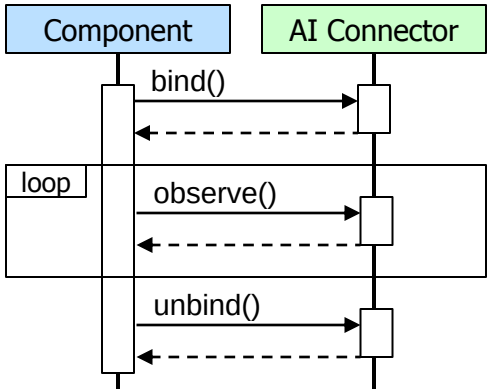
Interoperability Interface: Publishes ObservableSchema DDS message type that defines the connector, and interfaces with Monitoring Application via control topics (for remote control of AI)

- **API Allows User To:**
  - Bind an instance of the DDS message struct that types the AI connector
  - Take observation samples as desired (from stack, heap, executor…)
  - Unbind the struct before it goes out of scope (e.g., to bind another message struct instance)

- **Easy to Use:**
  - Add an AI port to a component in your MDA tool, including message struct type bound to it
  - Instantiate component & connector, and connect in an assembly
  - Use MDA tool to add instrumentation fields of interest to message
  - Set connector properties in deployment planning tool (use default DDS domain_id) – one time setting
  - Gen IDL & boilerplate executor, add API calls to appropriate callback stubs, build, deploy, view/control via Monitoring App at run-time



Basic CBDDS AI API



Discovery Connector

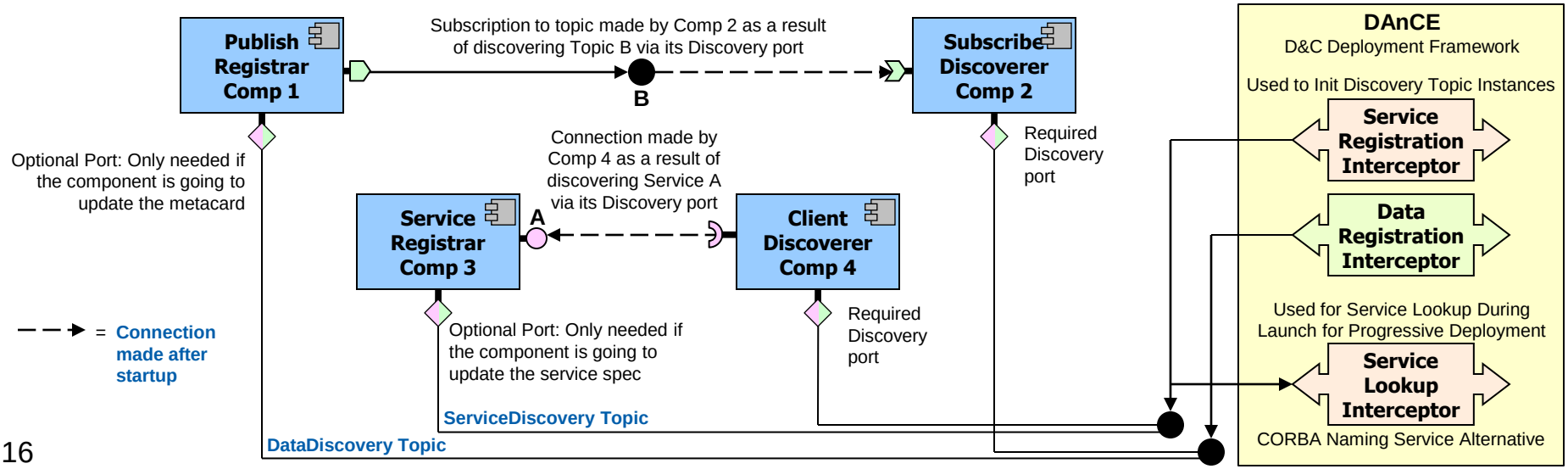
NORTHROP GRUMMAN

- Real-Time SOA Data & Service directory services access
  - Dynamic system behavior via run-time data and service registration, lookup & binding
- Common connector design typed for *either* data or service discovery (per port)
  - Data registry implemented as a keyed DDS topic
    - Enables registration and lookup of other topics (DDS built-in-topic extensions)
    - Topic metacard defined as an IDL subset of DDMS (DoD Discovery Metacard Specification) XML schema
  - Service Discovery registry implemented as a keyed DDS topic
    - Service spec metacard enables service endpoint registration & lookup
  - Both topics pre-defined by connector, with RELIABLE, KEEP_ALL, TRANSIENT_LOCAL QoS
- Discovery connector reuses mix of basic ports from the DDS_State connector
  - Encapsulates both subscribe and query, for lookup/discovery, and publish, for metacard updates & adds
- Supplemented by supporting DAnCE “interceptors”
  - DAnCE is an open source implementation of the OMG D&C spec execution model
  - Interceptors initialize DDS metacard instances during launch from CDP deployment plan tags
  - Service endpoint lookup during progressive deployment to establish connections with service components already deployed in a prior launch
    - Distributed (no single point of failure) alternative to the CORBA Naming Service

Data & Service Discovery

Initialization & Use Example

- At startup, DAnCE Registration Interceptors populate the initial instance messages in common ServiceDiscovery & DataDiscovery topics
  - Example: Service A is registered on the “SNA::ServiceDiscovery” topic on domain ID 17
  - Example: Topic B is registered on the “SNA::DataDiscovery” topic on domain ID 18
- Registrar components (Example Comp 1 & 3) have the *option* of defining Discovery connector ports & using them any time during system operation to update the instance messages originally created by the interceptors at startup
  - If they have no value to add or additional fields to fill in, these ports are not required
  - Discovery port type offers bidirectional read/write access to its underlying topic types (service or data discovery topic)
- Discoverer components (Example Comp 2 & 4) must use their Discovery ports after startup to lookup the service port or data topic on the other end to connect to, and then make API calls to connect (◂– –)



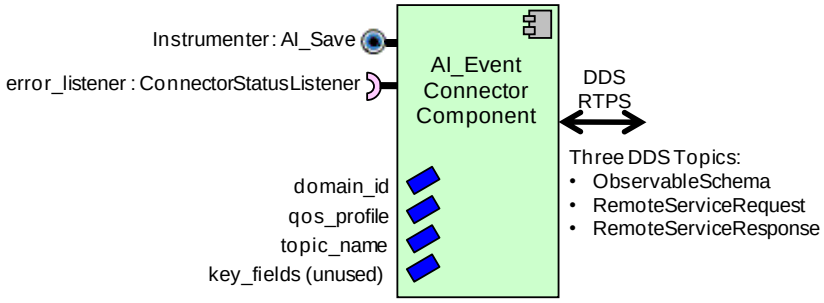
Ref: AI_Event and Discovery_State Fragments

Graphical Representation From ccm_ai.idl & ccm_discovery.idl

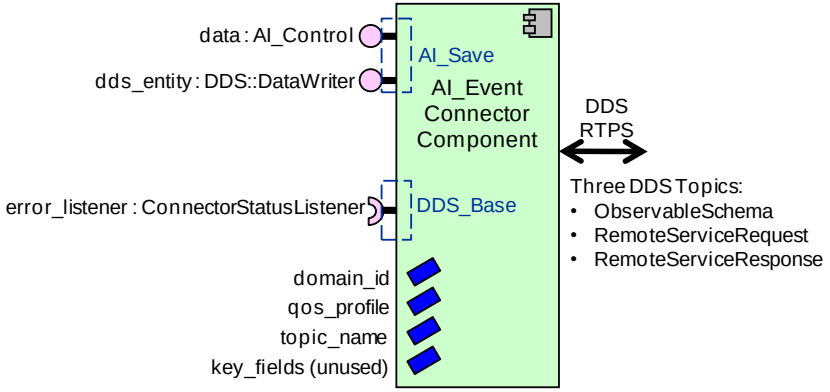
NORTHROP GRUMMAN

AI_Event

Extended Port View

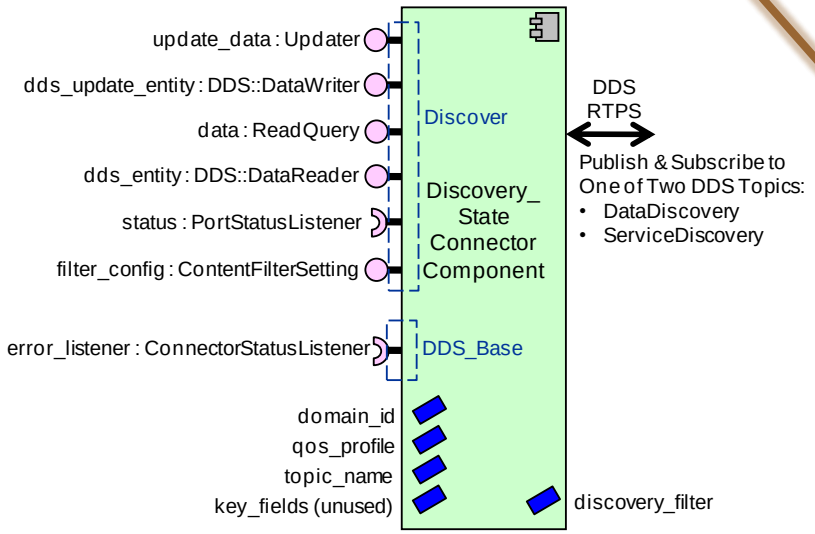


Basic Port View

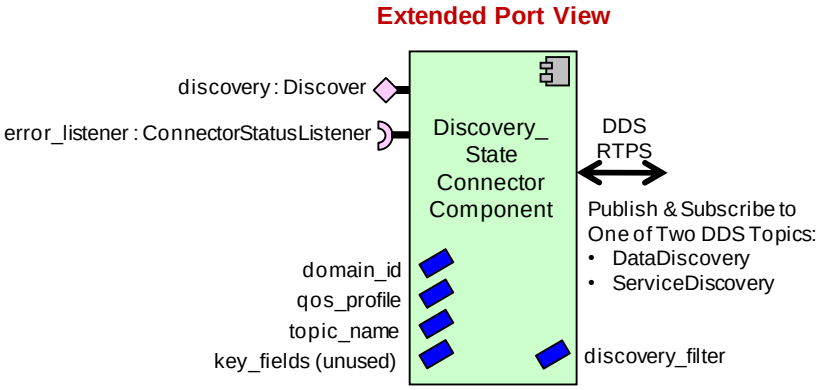


Discovery_State

Basic Port View



Extended Port View



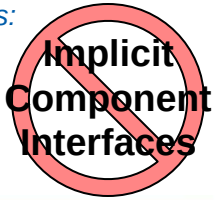
Connectors Can Encapsulate Just About Anything

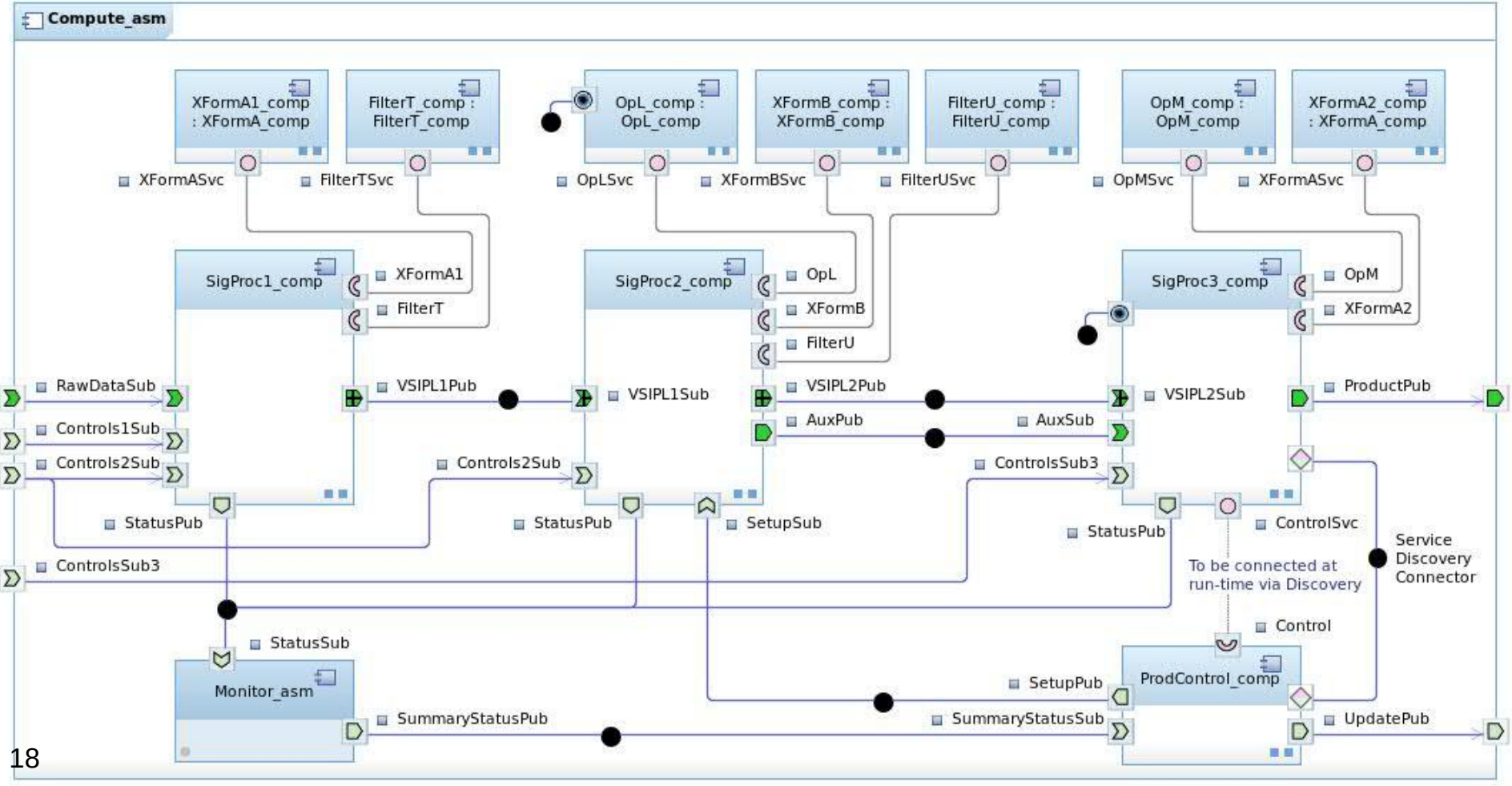
NORTHROP GRUMMAN

Easy Component Framework Integration, Well-Defined Interfaces

Useful for Building Bridge/Adapter Components With Ports that Adapt DDS or CORBA to Non-Native, External Middleware/Interfaces:

- TCP or UDP sockets
- Custom Shared Memory interfaces
- Serial ports
- Widely used legacy buses or device drivers
- Proprietary middleware (transition)
- JMS
- MPI
- D-Bus
- REST
- SOAP
- ZeroMQ
- Many, many more…





THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN

THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN

Backup Slides

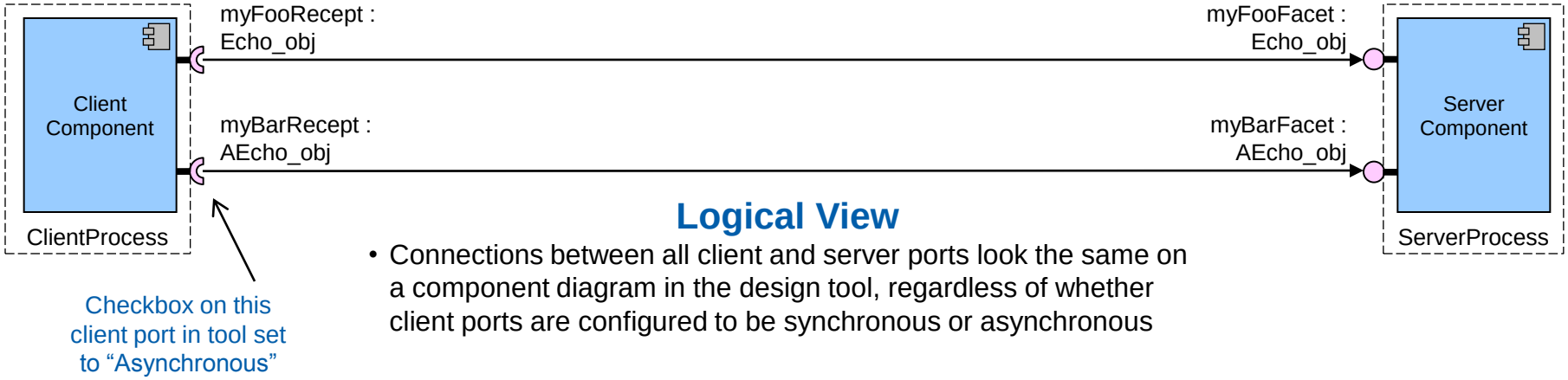
Abstract

In addition to defining normative connector types for integration of DDS middleware into a CCM component framework, the OMG DDS for Lightweight CCM (DDS4CCM) standard combined with recent CCM GIS improvements offers domain users the ability to define their own new or extended connector types. User-defined connectors can encapsulate any middleware or transport layer alternative to DDS, either extending DDS to enable enhanced functionality, or to replace it entirely with something else. By leveraging a generic, extensible GIS connector development framework, such as that provided by the DOC Group's CIAO product, users can build their own custom connector types to target a variety of domain specific applications. Moreover, extensible connector capabilities incorporated into Component Based DDS (CBDDS) UML profiles available from commercial MDA tool vendors Atego and Zeligsoft, enable users to build their own connector model libraries, making the use of custom connector types even easier.

This presentation will discuss four connector types that have been developed for Northrop Grumman's Teton Scalable Node Architecture (SNA) Platform, which is built upon a CBDDS application framework foundation. These custom connectors all extend the functionality of the normative DDS4CCM connectors. The first two connector types include a Publish Subscribe Attachment Transfer (PSAT) connector, for high-performance, zero/single copy shared memory or OpenFabrics Enterprise Distribution (OFED) based RDMA wideband data transport, as well a Signal Processing Data Model (SPDM) connector that extends PSAT to support the location independent transport of OMG VSIPL or VSIPL++ standard "views" for high-performance, component-based signal and image processing applications. The third connector type is a Discovery connector, which offers a DDS-only service registration and lookup, directory services alternative to the CORBA Naming Service for both D&C deployment framework "progressive" deployment as well as application-level DRE service discovery. The fourth connector type is an encapsulation of the still-in-development OMG Application Instrumentation (AI) standard for DDS-based binary data instrumentation, to supplement distributed, text-based logging.

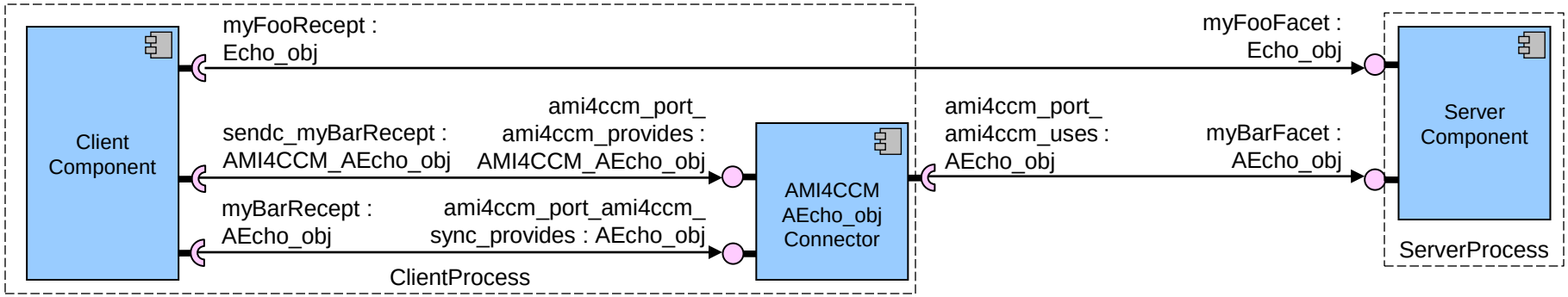
AMI4CCM Deployment

Connector Fragment & Connections



Logical View

- Connections between all client and server ports look the same on a component diagram in the design tool, regardless of whether client ports are configured to be synchronous or asynchronous



Concrete Deployment View

- Shows AMI4CCM connector fragment component actually deployed to encapsulate CORBA AMI code
  - Client synchronous calls actually made locally, and routed through the AMI4CCM connector as a pass-through
  - Enables future alternative technology connector implementations of both the sync & async calls from the client
- Shows implicitly created async client port and interface added by the IDL compiler to handle AMI sendc_* versions of calls
- Actual number of <connection> entries in CDP file between the AEcho_obj client and server ports is THREE (2 local, 1 normal), despite only 1 shown in Logical View
- UCM request-reply equivalent expected to add fragments on both ends (using either CORBA AMI or RPC4DDS)