Product Copier

Component Design Document

1 Description

Given two locations and a list of source/destination IDs, fetches Data_Product entries from one location and sends/copies them to another upon receiving a Tick.
A typical use case is for the two locations to be databases (e.g. Product_Database instances), and for this component to take snapshots of products in a source database at a fixed interval. While values stored in the source database may constantly be in flux, the destination database could provide a stable view of the source – within a tick, the values in the destination database will not change between reads.

2 Requirements

The requirements of the component.

1. The component shall copy data products from one location to another, every time it is sent a Tick, given a list of source ID/destination ID mappings.
2. The component shall fail at initialization when two mappings share the same destination ID.
3. If fetching from the source results in the data product not being available or the requested ID is out of range, no data product will be copied, and execution will continue. It will raise an error event if configured to do so.

3 Design

3.1 At a Glance

Below is a list of useful parameters and statistics that give a quick look into the makeup of the component.

- **Execution** - *passive*
- **Number of Connectors** - 5
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 4
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - *None*
- **Number of Parameters** - *None*
- **Number of Events** - 2

- **Number of Faults** - *None*
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

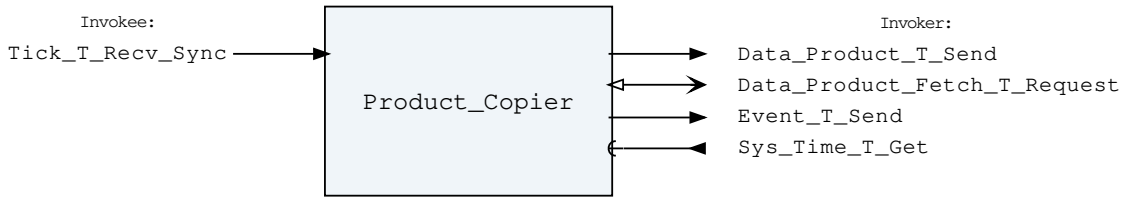

Figure 1: Product Copier component diagram.

3.3 Connectors

Below are tables listing the component's connectors.

3.3.1 Invokee Connectors

The following is a list of the component's *invokee* connectors:

Table 1: Product Copier Invokee Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Recv_Sync	recv_sync	Tick.T	-	1

Connector Descriptions:

- **Tick_T_Recv_Sync** - Triggers copying of data products (through request and send connectors).

3.3.2 Invoker Connectors

The following is a list of the component's *invoker* connectors:

Table 2: Product Copier Invoker Connectors

Name	Kind	Type	Return_Type	Count
Data_Product_T_Send	send	Data_Product.T	-	1
Data_Product_Fetch_T_Request	request	Data_Product_Fetch.T	Data_Product_Return.T	1
Event_T_Send	send	Event.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **Data_Product_T_Send** - The destination for fetched data products to be sent to.
- **Data_Product_Fetch_T_Request** - Where data products are copied from.
- **Event_T_Send** - Any produced events are sent out this connector.

- **`Sys_Time_T_Get`** - Time stamps for events are fetched via this connector.

3.4 Interrupts

This component contains no interrupts.

3.5 Initialization

Below are details on how the component should be initialized in an assembly.

3.5.1 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.5.2 Component Base Initialization

This component contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

3.5.3 Component Set ID Bases

This component contains commands, events, packets, faults, or data products that require a base identifier to be set at initialization. The `set_Id_Bases` procedure must be called with the following parameters:

Table 3: Product Copier Set Id Bases Parameters

Name	Type
`Event_Id_Base`	`Event_Types.Event_Id_Base`

Parameter Descriptions:
- **`Event_Id_Base`** - The value at which the component's event identifiers begin.

3.5.4 Component Map Data Dependencies

This component contains no data dependencies.

3.5.5 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. At initialization, this component requires a list of source/destination pairs of data products to copy. The `init` subprogram requires the following parameters:

Table 4: Product Copier Implementation Initialization Parameters

Name	Type	Default Value
`Products_To_Copy`	`Product_ Mapping_Array_ Access`	*None provided*
`Send_Event_On_Source_Id_Out_Of_Range`	`Boolean`	`True`
`Send_Event_On_Source_Not_Available`	`Boolean`	`False`

Parameter Descriptions:
- **`Products_To_Copy`** - The list of mappings to be copied by this component every tick. Raises an error on Init if the list is null, as well as if two mappings share a destination.

- **`Send_Event_On_Source_Id_Out_Of_Range`** - When the status of a fetch is of Id_Out_Of_Range, specifies whether an error event should be sent. This could indicate misconfiguration, so sending error events is the default.
- **`Send_Event_On_Source_Not_Available`** - When the status of a fetch is of Not_Available, specifies whether an error event should be sent. This might simply indicate that the product is not yet ready to be fetched, in which case this is expected behavior. Accordingly, not sending error events is the default.

3.6 Commands

The Product Copier component has no commands.

3.7 Parameters

The Product Copier component has no parameters.

3.8 Events

Below is a list of the events for the Product Copier component.

Table 5: Product Copier Events

Local ID	Event Name	Parameter Type
0	Source_Not_Available	Product_Copier_Error_Info.T
1	Source_Id_Out_Of_Range	Product_Copier_Error_Info.T

Event Descriptions:

- **`Source_Not_Available`** - A data product fetch resulted in an a Not_Available status, and was not sent to the destination.
- **`Source_Id_Out_Of_Range`** - A data product fetch resulted in an an Id_Out_Of_Range status, and was not sent to the destination.

3.9 Data Products

The Product Copier component has no data products.

3.10 Data Dependencies

The Product Copier component has no data dependencies.

3.11 Packets

The Product Copier component has no packets.

3.12 Faults

The Product Copier component has no faults.

4 Unit Tests

The following section describes the unit test suites written to test the component.

4.1 *Product_ Copier_ Tests* Test Suite

This is a set of unit tests for the Simple_Package package.

Test Descriptions:

- **`Test_Dest_Conflict`** - Tests whether two conflicting destinations raise an error.
- **`Test_Nominal_Tick`** - Tests the fetch and send operations caused by a tick.
- **`Test_Fetch_Fail_Behavior`** - Tests that no data products are sent to the destination when a fetch fails.
- **`Test_Fetch_Fail_Event`** - Makes sure an event is raised when a fetch operation fails and the corresponding init flag is set.

4.2 *Product_ Copier_ Tests* Test Suite

This is a set of unit tests for the Simple_Package package.

Test Descriptions:

- **`Test_Dest_Conflict`** - Tests whether two conflicting destinations raise an error.
- **`Test_Nominal_Tick`** - Tests the fetch and send operations caused by a tick.
- **`Test_Fetch_Fail_Behavior`** - Tests that no data products are sent to the destination when a fetch fails.
- **`Test_Fetch_Fail_Event`** - Makes sure an event is raised when a fetch operation fails and the corresponding init flag is set.

5 Appendix

5.1 Preamble

This component contains the following preamble code. This is inline Ada code included in the component model that is usually used to define types or instantiate generic packages used by the component. Preamble code is inserted as the top line of the component base package specification.

```
1 type Product_Mapping_Array is array (Natural range <>) of Product_Mapping.T;
2 type Product_Mapping_Array_Access is access all Product_Mapping_Array;
```

5.2 Packed Types

The following section outlines any complex data types used in the component in alphabetical order. This includes packed records and packed arrays that might be used as connector types, command arguments, event parameters, etc..

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 6: Data_Product Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length

Header	Data_Product_ Header.T	-	88	0	87	-
Buffer	Data_Product_ Types.Data_ Product_ Buffer_Type	-	256	88	343	Header.Buffer_ Length

Field Descriptions:

- **Header** - The data product header
- **Buffer** - A buffer that contains the data product type

Data_Product_Fetch.T:

A packed record which holds information for a data product request.

Table 7: Data_Product_Fetch Packed Record : 16 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	0	15

Field Descriptions:

- **Id** - The data product identifier

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 8: Data_Product_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	64	79
Buffer_Length	Data_Product_ Types.Data_Product_ Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the data product item.
- **Id** - The data product identifier
- **Buffer_Length** - The number of bytes used in the data product buffer

Data_Product_Return.T:

This record holds data returned from a data product fetch request.

Table 9: Data_Product_Return Packed Record : 352 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
The_Status	Data_Product_Enums.Fetch_Status.E	0 => Success 1 => Not_Available 2 => Id_Out_Of_Range	8	0	7	-
The_Data_Product	Data_Product.T	-	344	8	351	-

Field Descriptions:

- **The_Status** - A status relating whether or not the data product fetch was successful or not.
- **The_Data_Product** - The data product item returned.

Event.T:

Generic event packet for holding arbitrary events

Table 10: Event Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Event_Header.T	-	88	0	87	-
Param_Buffer	Event_Types.Parameter_Buffer_Type	-	256	88	343	Header.Param_Buffer_Length

Field Descriptions:

- **Header** - The event header
- **Param_Buffer** - A buffer that contains the event parameters

Event_Header.T:

Generic event packet for holding arbitrary events

Table 11: Event_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Event_Types.Event_Id	0 to 65535	16	64	79
Param_Buffer_Length	Event_Types.Parameter_Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the event.
- **Id** - The event identifier
- **Param_Buffer_Length** - The number of bytes used in the param buffer

Product_Copier_Error_Info.T:

To be sent in error events, specifying which source, destination, and tick caused the error.

Table 12: Product_Copier_Error_Info Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Tick	Interfaces. Unsigned_32	0 to 4294967295	32	0	31
Mapping	Product_Mapping.T	-	32	32	63

Field Descriptions:

- **Tick** - The count of the tick that caused the error.
- **Mapping** - *No description provided.*

Product_Mapping.T:

A source/destination ID pair that specifies where data products should be copied to/from

Table 13: Product_Mapping Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	0	15
Destination_Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	16	31

Field Descriptions:

- **Source_Id** - *No description provided.*
- **Destination_Id** - *No description provided.*

Sys_Time.T:

A record which holds a time stamp using GPS format including seconds and subseconds since epoch (1-5-1980 to 1-6-1980 midnight).

Table 14: Sys_Time Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Seconds	Interfaces. Unsigned_32	0 to 4294967295	32	0	31
Subseconds	Interfaces. Unsigned_32	0 to 4294967295	32	32	63

Field Descriptions:

- **Seconds** - The number of seconds elapsed since epoch.
- **Subseconds** - The number of 1/(2^32) sub-seconds.

Tick.T:

The tick datatype used for periodic scheduling. Included in this type is the Time associated with a tick and a count.

Table 15: Tick Packed Record : 96 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Count	Interfaces. Unsigned_32	0 to 4294967295	32	64	95

Field Descriptions:

- **Time** - The timestamp associated with the tick.
- **Count** - The cycle number of the tick.

5.3 Enumerations

The following section outlines any enumerations used in the component.

Data_Product_Enums.Fetch_Status.E:

This status denotes whether a data product fetch was successful.

Table 16: Fetch_Status Literals:

Name	Value	Description
Success	0	The data product was returned successfully.
Not_Available	1	No data product is yet available for the provided id.
Id_Out_Of_Range	2	The data product id was out of range.