



Non-Transient Incident Handles in C

Version 1.2
December 31, 2008

PathMATE Technical Notes

Pathfinder Solutions
33 Commercial Drive, Suite 2
Foxboro, MA 02035 USA
www.PathfinderMDA.com
+1 508-568-0068

Table Of Contents

1. Introduction	1
2. Usage	1
3. Implementation	1
Mechanisms	1
Templates	2
4. Implementation Notes	2

1. Introduction

In the PathMATE C Map, incident handle (service handle) delivery is accomplished by doing a deep copy of the service handle structure, including data container parameters, dispatching the service handle for handling, then deleting the copy. The creation, population and deallocation of these transient copies are materially inefficient when the service handle is called very frequently, such as with recurring signals or periodic services. The Nontransient Incident Handle feature provides a way to avoid this copy-on-send overhead.

It has been observed that frequently used service handles are stored as attributes of a class. To reduce the overhead of processing these service handles, the attribute holding the service handle can be marked with a **NonTransient** property, set to **T** or **F**. When set to **T**, the service handle will be dispatched directly, and not copied and deleted.

Only attributes can be marked as containing non-transient service handles. Periodic services and local variables containing service handles will not be optimized.

2. Usage

A service handle may be marked **NonTransient** by marking the attribute to hold the non-transient service handle.

Use the *properties.txt* file to mark the service model element. For example:

```
# Non-transient marking for a frequently used service handle
Attribute,System.Domain.className.attrName,NonTransient,T
```

These non-transient services handles are then invoked directly, rather than through the default implementation that creates a copy for dispatch. The service handle is cleaned up (deleted) when an object containing the service handle as an attribute is deleted.

As a side effect of this feature, parameter values passed to service handles are "sticky". Parameters set for one call would still be set on the next, even if not set then.

3. Implementation

This feature is implemented by modifying the `SW_Incident_deallocate()` function and not truly deleting it if the handle type is `SW_NONTRANSIENT_SERVICE_HANDLE`. Dispatching code also checks this type to determine if a copy should be sent or the original.

Non-transient service handles require a number of changes to templates and mechanisms to support.

Mechanisms

Files associated with `SW_Incident` and `SW_Task` have been changed to support a new non transient type as the `SW_Incident` type,

SW_NONTRANSIENT_SERVICE_HANDLE. Management of this type, proper creation, deletion, and dispatching is implemented here.

Templates

Templates to check the **NonTransient** property on creating and dispatching the service handle are updated. Templates implementing the class destructor are updated to handle cleaning up these service handles, as well as attribute assignments.

4. Implementation Notes

A more general approach than what is specified in this document was tried – essentially allowing incident handles of any form (not just attributes) to be marked as non-transient - and numerous complication and inefficiencies developed. The current approach was selected because of the commonality of the pattern, and for its simplicity.

Files changed – mechanisms:

- sw_incident.c/h
- sw_pool_block.c
- sw_task.c/h

Files changed – templates:

- act_stmt_assignment.arc
- act_stmt_invoke_sh.arc
- obj_create_guts.arc
- obj_ctor.arc