



PathMATE Maps IncidentHandleProfile

Version 1.2
April 4, 2004

PathMATE Technical Notes

Pathfinder Solutions LLC
33 Commercial Drive, Suite 2
Foxboro, MA 02035 USA
www.PathfinderMDA.com
888-662-7284

Table Of Contents

- 1. **Introduction** 1
- 2. **General Usage** 1
 - Example.....1
- 3. **IncidentHandleProfile** 2
 - Definition2
 - Reference.....2
 - Example.....3
 - Definition*3
 - Reference*.....3
 - Rules/Conventions.....3
- 4. **Profile Property Management**..... 3
- 5. **Feature Implementation** 3

1. Introduction

This Technical Note describes a mechanism for defining parameter Profiles for IncidentHandles. Profiles will enable stringent checking of IncidentHandle parameter accesses, and reduce the unnecessary system-level compile-time turbulence caused by parameter addition/deletion/renaming within domains.

An IncidentHandle Profile is a named property set that defines the number, name and datatype of parameters that may be accessed in PAL contexts outside of the defining context. An IncidentHandle's defining context is the action it is created within. If an attempt is made to access a parameter of an IncidentHandle that is not "published" via its IncidentHandle Profile, a translation-time message will result in the generated code, preventing successful compilation.

2. General Usage

In general the usage of IncidentHandles does not require the CALLing context to set any parameter values. In these cases, it is sufficient to simply CALL the IncidentHandle, and no additional information about the IncidentHandle – specifically the parameters that may be published by the based service (the service pointed to by the IncidentHandle). In these cases, no parameter information is needed in the calling context.

An IncidentHandle profile is used whenever a recipient (generally the CALLing context) of an IncidentHandle needs to write a parameter value to that IncidentHandle. A common example of this is when a client domain requests an activity to be done within a server domain, and requires a callback with a status parameter be sent when the activity is done. The server domain needs to write the status parameter to the callback IncidentHandle before it is CALLED.

Example

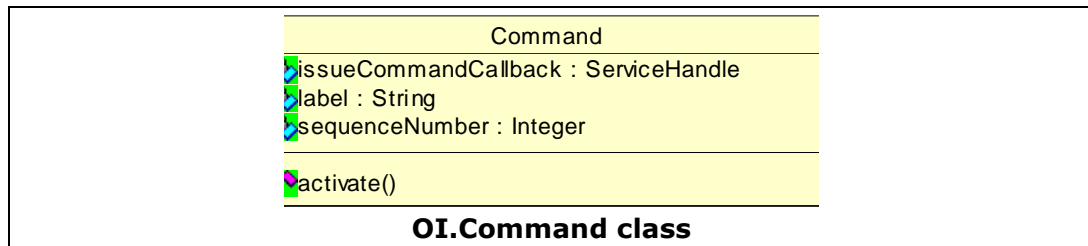
In the CarShuffle sample system, a somewhat different case is shown. The VehicleHousing(VH) domain registers user input callbacks with the OperatorInterface(OI) domain via the OI:RegisterWithContextCommand domain service. These user input callbacks require an `input` parameter to convey the user input, and a `client_callback` parameter to be called when the VH activity resulting from the input completes. In this case we will focus on the handler for the "Move In" command. In VehicleHousing, the Car:moveIn class service accepts user "Move In" commands:

C:moveIn(instance-based): *Moves the car into the specified garage. This is registered with OI as a callback requiring the "input" and "client_callback" parameters.*

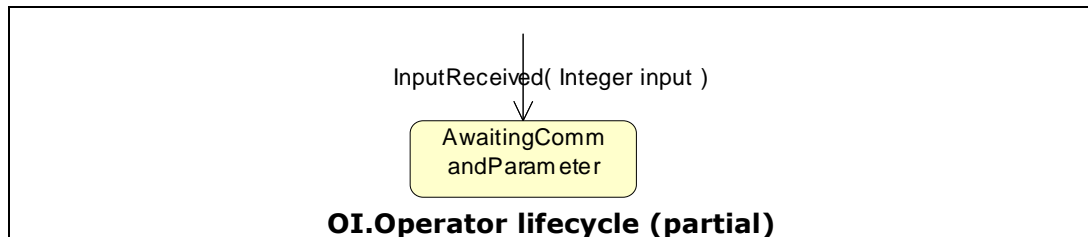
in: **input(Integer):** *Indicates which garage to move into.*

in: **client_callback(IncidentHandle)**

During start up an IncidentHandle to C:moveIn is created and passed into OI:RegisterWithContextCommand, where it ends up stored in the Command.issueCommandCallback attribute:



When the user issues the "Move In" command and then provides the target garage number, the OI.Operator.AwaitingCommandParameter entry action sends the garage number back to VH:



OI.Operator.AwaitingCommandParameter entry action (partial):

```
CALL cmd.issueCommandCallback(input = input, client_callback =
command_complete);
```

Once we define how to define an IncidentHandleProfile, we will return to this example and apply the profile to it.

3. IncidentHandleProfile

Definition

Base Incident: Every IncidentHandle that allows any of its parameters to be written in PAL contexts outside of its defining context must define an IncidentHandleProfile. This is done by specifying the IncidentHandleProfile name via the "Profile" property of the IncidentHandle's *base incident* - via the base incident's Rose Specification PathMATE tab, or via properties.txt. The *base incident* is the domain or class service that is referenced with the IncidentHandle is created.

Profile: By default, all parameters defined for the base incident are published. In the case where some parameters are to be published and others are not, the published parameters explicitly set their property "ProfileParameter" to "T" - via the property's Rose Specification PathMATE tab, or via properties.txt.

Reference

When an IncidentHandle parameter is to be written outside of its defining context, the data atom carrying that IncidentHandle must refer to the IncidentHandleProfile publishing that parameter. This is done by specifying the IncidentHandleProfile name via the "Profile" property at the point of declaration:

- For class attributes, the IncidentHandleProfile name is specified via a "Profile" property of the attribute.

- For service or event parameters, the IncidentHandleProfile name is specified via a "Profile" property of the parameter.
- For action local variables, the IncidentHandleProfile name is specified via a "Profile" property of the local variable declaration statement.

Example

The CarShuffle sample system includes the use of Profiles.

Definition

A profile is defined for the user input callback. The Car:moveIn class service property "Profile" is set to "InputWithParam". The parameters `input` and `client_callback` are published by default.

Reference

The "InputWithParam" service handle parameters `input` and `client_callback` are written to the `Command.issueCommandCallback` attribute in the `OI.Operator.AwaitingCommandParameter` entry action. Therefore we set the `Command.issueCommandCallback` attribute property "Profile" to "InputWithParam".

Rules/Conventions

- A single named IncidentHandleProfile may be defined by more than one base incident. In this case, the names and datatypes of all published parameters must match.
- The "this" parameter (destination) of an instance-based class service cannot be published.
- It is not necessary to reference the IncidentHandleProfile for a data atom that does not have parameters accessed from it, even if the service handle's base incident defines an IncidentHandleProfile. In the example above, the `OI:RegisterWithContextCommand` domain service parameter `callback` does not need to reference any IncidentHandleProfile.
- If a service has an IncidentHandleProfile and the service has one or more parameters, but no parameters have "ProfileParameter" == "T", all parameters of this service are assumed to be published as part of the profile.

4. Profile Property Management

Since an IncidentHandleProfile is a characteristic of the base analysis model, and is not a design-specific marking, the "Profile" and "ProfileParameter" parameters will be incorporated into the Rose-visible PathMATE property set, and set within Rose.

5. Feature Implementation

The goals of this feature are:

- Provide enforcement translation-time checks on the data integrity of IncidentHandle usage.

- Speed incremental compile time by eliminating inter-domain coupling introduced by the SYS_PARAMNUM enumerate.

This feature provides code generation templates (applied for C++, C, and Java generation) that maintain profile information for all IncidentHandleProfiles and correlate this with the profile usage where service handle parameters are accessed. Transformation-time checks are performed to ensure:

- IncidentHandle parameters can only be accessed from IncidentHandle data atoms that reference a pre-defined profile.
- The parameter names and data types for all such accesses must match the referenced profile.
- A service with no parameters may not specify an IncidentHandleProfile.
- (All Rules/Conventions in section 3.4 above.)

To reduce the unnecessary system-level compile-time turbulence caused by parameter addition/deletion/renaming within domains, the SYS_PARAMNUM enumerate has been eliminated. Instead each IncidentHandleProfile generates its own include file, containing a set of constants identifying each published parameter in the profile.

The profile include file is named `sys_profile_<profile name>.hpp` or `sys_profile_<profile name>.h` or `SysProfile<profile name>.java`. This profile definition file is only included in files where the profile's parameters are actually accessed. This profile definition file also contains a documentation summary of its profile, including name, defining service, and published parameter names and types.