



Platform-Independent Action Language Parsing Directives

Version 1.0
June 17, 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. Feature Overview..... 1**
 - Parsing Control1
 - Implementation Code Inline1
 - Conditional Inclusion2
 - Implementation Code #INCLUDE3
 - Rules/Conventions.....3
- 3. Feature Implementation 4**
 - Transformation Engine.....4
 - Design Changes4
- 4. MDA Philosophy Note..... 4**

1. Introduction

This Technical Note describes features to be provided in the PathMATE product family for specific PAL parsing directives to support implementation code flexibility and PAL action configurability. These goals will be achieved with the following PAL parser directives:

- Parsing control via ParseActions property for System, Domain, Subsystem, Class
- #INLINE and #END_INLINE with implementation code between
- #IFDEF *value* and #ENDIF where *value* is a system property
- #INCLUDE *file* that injects the contents of implementation code files into the generated code for an action

2. Feature Overview

Parsing Control

This feature allows the modeler control over which actions get parsed as valid PAL, and which actions are treated as completely implementation code and not parsed at all. By setting the ParseActions property (T, F; default T) either in Rose on the PathMATE tab or via properties.txt, all actions within the affected unit are controlled. This includes initialization actions for systems and domains as appropriate. This property is available for the System, Domain, Subsystem and Class.

If a modeler wishes to suppress the parsing of a single action, they may place a #INLINE directive at the top line of the action (see section 0 Implementation Code Inline below).

Implementation Code Inline

The #INLINE and #END_INLINE will allow implementation code to be placed within a PAL action. Upon transformation, the lines containing #INLINE and #END_INLINE will be discarded and the text between will be emitted directly in the target document. No PAL parsing or checking will be done on this text.

For example, consider the following action:

```
Integer item_count = 0;
#INLINE
    char *datap = (char*) &item_count;
    // Move up 1 byte
    datap += 1;
    // Pass off
    external_mystery_function(datap);
#END_INLINE
```

This action will transform into the following C implementation code:

```
int item_count = 0;
    char *datap = (char*) &item_count;
    // Move up 1 byte
    datap += 1;
    // Pass off
    external_mystery_function(datap);
```

Conditional Inclusion

System properties can be used to control the inclusion/exclusion of segments of PAL actions with `#IFDEF value` and `#ENDIF`. In addition, `#ELSE` and `#ELSE_IFDEF value` are also supported. If the value of the PathMATE System property value is not empty (""), then the PAL statements and directives between the `#IFDEF value` and the corresponding `#ENDIF` will be included. Otherwise they will be discarded.

For example, consider the following action:

```
Integer item_count = 0;
// Compute the sum based on the system type

#ifdef TruckBased
    FOREACH pallet = truck->A1
    {
        item_count = item_count + pallet.itemCount;
    }
#else_ifdef ShipBased
    FOREACH container = storage_building ->A2
    {
        FOREACH pallet = container->A3
        {
            item_count = item_count + pallet.itemCount;
        }
    }
#else
    FOREACH item = hand_cart->A4
    {
        item_count = item_count + 1;
    }
#endif // ShipBased
```

If the System property `TruckBased == "T"` and the System property `ShipBased == ""`, the above action will reduce to:

```
Integer item_count = 0;
// Compute the sum based on the system type

    FOREACH pallet = truck->A1
    {
        item_count = item_count + pallet.itemCount;
    }
```

If the System property `TruckBased == ""` and the System property `ShipBased == "T"`, the above action will reduce to:

```
Integer item_count = 0;
// Compute the sum based on the system type

    FOREACH container = storage_building ->A2
    {
        FOREACH pallet = container->A3
        {
            item_count = item_count + pallet.itemCount;
        }
    }
```

If the System property `TruckBased` and `ShipBased` both `== ""` the above action will reduce to:

```
Integer item_count = 0;
// Compute the sum based on the system type

    FOREACH item = hand_cart->A4
    {
        item_count = item_count + 1;
    }
```

Implementation Code #INCLUDE

The contents of implementation code files can be injected into the generated code for an action. The PAL parser directive `#INCLUDE file` will look in the `includes` subdirectory (relative to the current working directory of the code generation activity) and if the specified file is found, its contents will be inserted directly where the `#INCLUDE` directive was.

For example, consider the following action:

```
Integer item_count = 0;
#include middle.c
```

Assume the file `project/c/include/middle.c` exists and this file contains:

```
printf("Item count = %d.\n", item_count);
```

The above action will expand into the following implementation code if translated to C:

```
int item_count = 0;
printf("Item count = %d.\n", item_count);
```

To facilitate support in implementation languages where there is no `#include` directive, such as Java, the actual reading of the file will happen at transformation time.

Rules/Conventions

- A directive is expected to start in the first character of its line – “#” in column 1.
- All directives are expected to be the only thing on the line of text containing them (except for comments) – they cannot be co-located with PAL statements on the same line.
- Upon transformation all directives will be discarded and only their results will be available in the target document
- `#INLINE` and `#END_INLINE` directives must be matched. Encountering an `#INLINE` within an already-`INLINE` section will result in an error. Encountering an `#END_INLINE` outside of an `INLINE` section will result in an error. Encountering an end of file/action within an `INLINE` section will result in a warning.
- `#IFDEF` and `#ENDIF` directives must be matched. Encountering an `#ENDIF` outside of an `IFDEF` section will result in an error. Encountering an end of file/action within an `IFDEF` section will result in an error.

- Any PAL parser directives within a `#INLINE` section will be ignored except for `#END_INLINE`.
- A `#INCLUDE` directive with a blank *file* will result in an error.

3. Feature Implementation

Transformation Engine

The Engine PAL parser will need to be changed to identify and properly handle these new directives. A new Statement type `ImplementationCode` will be created with a String field `rawText`. An instance of `ImplementationCode` will be created for each line of text appearing between the `#INLINE` and `#END_INLINE` directives. In addition `#INCLUDE` directives will be converted into an instance of `ImplementationCode` where the `rawText` value will be the contents of the included file.

Design Changes

The Maps will have to handle a new the PAL statement type – `ImplementationCode` – emitting the `rawText` directly.

4. MDA Philosophy Note

Model Driven Architecture brings many benefits to its practitioners. Many of these benefits – and the most important of these benefits – are derived from the key principle of separation of the Platform Independent Models from specific implementation details. The interleaving of implementation code with PAL in actions is a very direct and insidious violation of this separation, and thereby seriously undermines the effectiveness of such models. It is Pathfinder Solutions' position that the `#INCLUDE`, `#INLINE` and `#END_INLINE` directives should only be applied in the most judicious manner, and be treated as potentially dangerous. Good luck.