



---

## **Index-Based Instance Identification**

Version 1.1  
December 27, 2006

---

### *PathMATE Technical Notes*

Pathfinder Solutions LLC  
33 Commercial Drive, Suite 2  
Foxboro, MA 02035 USA  
[www.PathfinderMDA.com](http://www.PathfinderMDA.com)  
888-662-7284

# Table Of Contents

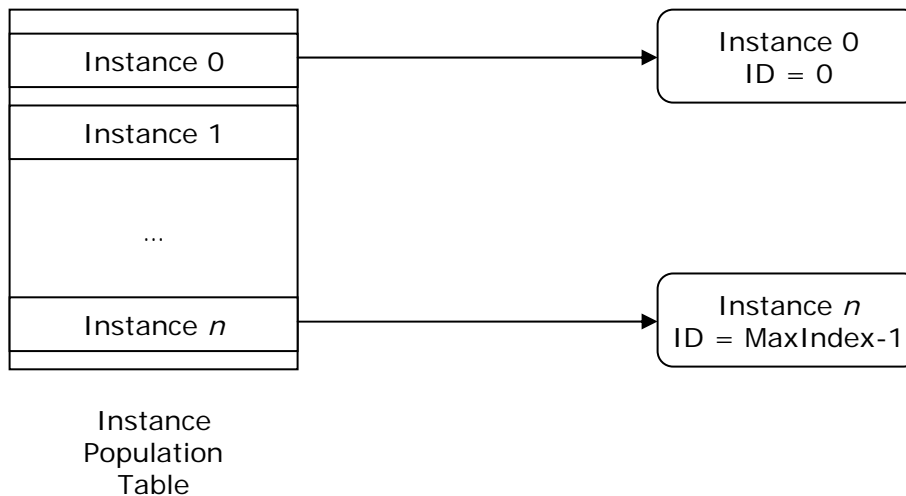
- 1. **Introduction**..... 1
- 2. **Usage** ..... 1
- 3. **Implementation** ..... 2
  - Instance creation ..... 2
  - Class-based finds ..... 2
  - Instance deletion ..... 2
- 4. **Implementation Notes** ..... 3

## 1. Introduction

Index-based instance identification allows for the efficient retrieval of object instances from an instance population. Object instances using index-based instance identification contain an “identifier” attribute - indicated by the **IndexID** marking - that uniquely identifies the instance. This marking will result in the generation of an instance population structure that uses the C++ Standard Template Library `std::map`, reducing FIND/WHERE constructs based on the id attribute to map lookup via `std::map::find()`.

Alternatively when used in conjunction with the **MaxIndex** class marking, this will result in the generation of an instance population structure using an array, reducing FIND/WHERE constructs based on the id attribute to a direct array element access.

The following diagram show how the index-based instance population management manages the instance table. The class is defined to have an attribute named “ID” which is marked as the instance identifier. When instances of the class are created, the value of the “ID” attribute is used to determine the location in the population table where the instance will reside.



When PAL FINDs are used to locate an instance using the “ID” attribute in a where clause, the implementation can perform a rapid index-based retrieval of the object rather than performing the usual linear table search strategy.

## 2. Usage

To enable index-based instance identification, one of the class’ attributes must be marked with the **IndexID** value set to “TRUE”. Normally, this marking takes place in the *properties.txt* file during transformation.

Optionally the class can use the **MaxIndex** marking to enable array-based instance management.

Transformation-time errors are logged using the Transformation Engine's LOG\_MESSAGE statement in the following cases:

- More than one attribute is marked with the **IndexID** marking.
- The attribute marked with the **IndexID** marking is not typed as an integer.
- An attribute is marked with **IndexID** but the **MaxIndex** marking is not enabled.

At runtime, it is an error to:

- Create an instance whose index is negative or greater than the limits imposed by the **MaxIndex** marking.
- Create multiple instances with the same index value.

## 3. Implementation

Index-based instance identification affects several areas of the design including:

- how instances plug themselves into the instance population set.
- how WHERE clauses are processed when the index attribute is specified in the clause
- how instances unplug themselves from the instance population

### New Mechanism - PfdInstanceMap

For non-array based storage the PfdInstanceMap class is used. This class encapsulates the usage of the `std::map`.

### Instance creation - array

The instance "plug-in" handling for object instances changes when the **IndexID** marking is used. Without the index ID marking, an object instance is inserted into the first available slot in the table. With the **IndexID** marking, the instance is inserted at the offset specified by the value of the index attribute. If the slot is already occupied, the plug-in code should not overwrite the existing instance. With the **IndexID** marking, the instance table may be sparsely populated. Any iteration over the instance table must guarantee that empty entries are handled gracefully.

### Class-based finds

The generation of class-based finds will be modified to detect cases where the find operation can be optimized with index-based lookups for the instance population. Given the case where a find expression's "WHERE" clause contains only the **IndexID** attribute in an equivalence comparison, the implementation can efficiently retrieve the instance from the table without having to search the entire population. Additional optimization should be possible for where clauses that contain the **IndexID** attribute when used in conjunction with a logical-and operation.

### Instance deletion - array

The instance deletion ("unplug") handling must clear the object instance from its slot in the instance table. This must be done without altering the offsets of any subsequent entries in the instance population.

## 4. Implementation Notes

- Assignment to an attribute marked with the **IndexID** marking is not supported. This attribute should be treated as a constant and not changed over the lifetime of the instance.