## Analysis Report for: 1D1083180ED73BB2B88CCF24167E16F2.cs

**Overall Functionality**

The code consists of several files belonging to different projects. The core functionality revolves around a Dynamics NAV codeunit (`Codeunit7050260`) that manages change logs. This codeunit is triggered by an event (`OnAfterIsAlwaysLoggedTable`) and determines whether a specific table should always be logged. Other files contain assembly information, a custom attribute (`DotfuscatorAttribute`), and what appears to be configuration data for access rules. The code is obfuscated, using unusual variable names (e.g., αscopeId, γeventScope) and seemingly irrelevant calculations within `ConfigRegrasAcesso::CarregarConfiguracaoRegrasAcesso`.

**Function Summaries**

* **`Codeunit7050260::Codeunit7050260(ITreeObject parent)`:** Constructor for the codeunit. Initializes the base class with the parent object and the codeunit ID (7050260).
* **`Codeunit7050260::ObjectName`:** Getter property that returns the name of the codeunit ("GsiChangeLogManagement").
* **`Codeunit7050260::OnInvoke(int memberId, object[] args)`:** Handles method invocations on the codeunit. It checks the `memberId` and dispatches to the appropriate method (currently only `OnAfterIsAlwaysLoggedTable`). Throws an error if the member ID is unknown or the number of arguments is incorrect.
* **`Codeunit7050260::__Construct(ITreeObject parent)`:** Static factory method to create an instance of `Codeunit7050260`.
* **`Codeunit7050260::OnClear()`:** Empty override of the `OnClear` method. This method is likely called when the codeunit is being cleaned up or released.
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable(int tableID, ByRef alwaysLogTable)`:** This is the core logic. It's an event subscriber that determines whether a table should always be logged. It uses a nested scope (`OnAfterIsAlwaysLoggedTable_Scope`) to manage its execution.
* **`Codeunit7050260::IndirectPermissionList`:** Getter property that returns an array of indirect permission IDs.
* **`Codeunit7050260::()`:** Initializes the `indirectPermissionList` with four permission IDs.
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable_Scope::OnRun()`:** The main logic of the event handler. It checks several conditions related to a `changeLogSetup` record and sets `alwaysLogTable` to `true` if those conditions are met.
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable_Scope::RawScopeId`:** Getter/setter for a scope ID (likely used for internal Dynamics NAV management).
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable_Scope::EventScope`:** Getter/setter for an event scope (likely used for internal Dynamics NAV management).
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable_Scope::OnAfterIsAlwaysLoggedTable_Scope(Codeunit7050260 βparent, int tableID, ByRef alwaysLogTable)`:** Constructor for the nested scope class.
* **`DotfuscatorAttribute::DotfuscatorAttribute(string a, int c)`:** Constructor for a custom attribute, likely used for obfuscation.
* **`DotfuscatorAttribute::A`:** Getter for the 'a' field of the `DotfuscatorAttribute`.
* **`DotfuscatorAttribute::C`:** Getter for the 'c' field of the `DotfuscatorAttribute`.
* **`ConfigRegrasAcesso::CarregarConfiguracaoRegrasAcesso()`:** This function returns a configuration object containing a list of procedures (`ProcDLL`) and their associated enum aliases. The function contains obfuscated code that doesn't seem to affect the return value.

**Control Flow**

* **`Codeunit7050260::OnInvoke`:** A simple `if-else` statement checks the `memberId`. If it matches 7050000, it calls `OnAfterIsAlwaysLoggedTable`; otherwise, it throws an error. Argument validation is performed before calling `OnAfterIsAlwaysLoggedTable`.
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable`:** Creates an instance of `OnAfterIsAlwaysLoggedTable_Scope` and calls its `Run()` method within a `using` statement (ensuring proper disposal).
* **`Codeunit7050260::OnAfterIsAlwaysLoggedTable_Scope::OnRun`:** Contains a complex conditional statement (`if`) involving multiple checks using `CStmtHit` and calls to methods on `changeLogSetup.Target`. If the condition is true, `alwaysLogTable.Value` is set to `true`.

**Data Structures**

* **`uint[] indirectPermissionList`:** An array of unsigned integers representing permission IDs.
* **`Codeunit7050260.OnAfterIsAlwaysLoggedTable_Scope`:** A nested class acting as a scope for the event handler. It contains fields for the table ID, the `alwaysLogTable` boolean, and a `NavRecordHandle` named `changeLogSetup`. `NavRecordHandle` appears to be a class for managing access to records in the database.
* **`ConfiguracaoRegrasAcesso`:** A class (in `ConfigRegrasAcesso.cs`) that holds a collection of `ProcDLL` objects. `ProcDLL` seems to represent a procedure (likely stored procedure) and its associated parameters (enum aliases).

**Malware Family Suggestion**

Based solely on the provided code, it's difficult to definitively label it as belonging to a specific malware family. The code itself doesn't exhibit malicious behavior like network communication, file system manipulation, or data exfiltration. However, several aspects raise suspicion:

* **Obfuscation:** The heavy use of obfuscation techniques (unusual variable names, seemingly useless calculations) is a common tactic to hinder reverse engineering and analysis, often employed by malware authors.
* **Suspicious Logic:** The complex conditional statement in `OnAfterIsAlwaysLoggedTable_Scope::OnRun` could be used to perform checks that might be related to malicious activities if the `changeLogSetup` record's data and the hardcoded table ID (2000000073) are examined in detail within the Dynamics NAV context.
* **Unclear Purpose:** The overall purpose of manipulating the `alwaysLogTable` boolean within the Dynamics NAV system is unclear without more

context. This could potentially be used to covertly manipulate logging or auditing functionality.

Therefore, while the code is not inherently malicious, its obfuscation and potentially suspicious logic warrant further investigation in the context of its runtime environment. It is likely part of some backdoor or an advanced persistent threat that has been detected and is being studied. Further analysis of the `changeLogSetup` record and other Dynamics NAV objects would be required to determine its true nature. It's not possible to assign it to a specific malware family without this additional context.