Clousot Overview

# Overview

Clousot is the static checker for CodeContracts. It performs deep semantic analysis to discover ***facts*** about the program. It uses the inferred facts to report possible errors and ***to suggest program repairs***. In the default settings, the only bugs reported to the user are either provable or highly likely. This means programmers are not bothered by a large number of possible errors. Program analysis runs in the background and suggests code repairs at design time, *e.g.*, missing preconditions/parameter validation, equivalent non-overflowing expressions, correct variable initializations, stronger guards *etc*.

Clousot uses contracts to achieve modularity, scalability and to increase precision.

Clousot is widely tested (even on huge assemblies) and it is very stable. It is used within Microsoft as well as outside. It has been downloaded externally more than 50,000 times over the three years it has been available on DevLabs.

# Architecture

Clousot has 5 main phases: IL reading, fact discovery, assertion checking, report and repair suggestions, and inter-method propagation.

Analyses results, suggestions, inferred facts and program repairs are cached into an external SQL database. The database can be stored on a server and shared by team members.

### IL reading

Clousot begins with the IL generated by a .NET compiler. It uses a component called a *code provider* to access the IL of each method body, the contracts for each method (and object invariants for each type). It uses the code provider to construct a control-flow graph (CFG) of each method it analyzes. Clousot is totally parametric with respect to the code provider. Currently we have three code providers available (for CCI1, CCI2, and Roslyn) that implement the Clousot API.

Contracts are extracted (*e.g*., from contract assemblies) and attached so to create an annotated CFG. In the annotated CFG contracts are made explicit, as assertions or assumptions.

### Fact discovery

On the top of the annotated CFG, several static analyses are run.

A first set of analyses has the goal of providing a more abstract view of the program to simplify and factor out the task of the upper analyses. First the stack is made explicit and aliasing is resolved. Then, some of the high level structure lost in the compilation is reconstructed (to improve the precision of downstream analyses).

The fact discovery analyses are run on the result of the previous analyses. Examples of discovered facts are: which value is null/not-null, the numerical and the symbolical ranges for variables, the precise tracking of enum values, the content of arrays *etc.*

The fact discovery analyses in Clousot are at the state of the art. This Channel9 video details how they work (starting at minute 10):

<http://channel9.msdn.com/blogs/peli/static-checking-with-code-contracts-for-net>

###### Example. In the (incorrect) version of the Binary Search below, Clousot automatically discovers the loop invariant:

0 ≤ inf ≤ sup ≤ array.Length

static int BinarySearch(int[] array, int value)

{

var inf = 0;

var sup = array.Length;

while (inf <= sup)

{ // inferred fact: 0 ≤ inf ≤ sup ≤ array.Length

var index = (inf + sup) / 2;

var mid = array[index];

if (value == mid) return index;

if (mid < value) inf = index + 1;

else sup = index - 1;

}

return -1;

}

### Assertion Checking

The facts are used to prove program assertions. The user can choose which assertions should be checked. Assertions include contracts, non-null obligations, array accesses, integer overflows, buffer overruns in unsafe code, division by zero, negation of MinValue, floating point comparisons *etc.*

There are four possible outcomes for a checked assertion: *True,* the assertion is valid for all the program executions (correct); *False,* the assertion will fail every time the execution reaches it (definite error); *Unreached,* there is no execution reaching the assertion (dead code); *Unknown,* Some information is missing, or there exists some input for which the assertion is correct and some for which it is wrong (warning).

###### Example. In the binary search above, Clousot uses the inferred facts to prove the absence of null-dereferences, array out of bounds, and arithmetic overflows. It detects three possible bugs:

###### The input array may be null, so that the dereference of array may throw an exception;

###### The expression (inf + sup) may overflow (so index may be negative);

###### index ≤ array.Length, so that a buffer overrun may occur in the array load.

###### It proves that the other arithmetic operations do not overflow and that when the ldelem is executed, array is not null (otherwise an exception would have been thrown before)

### Report and Verified Repair suggestions

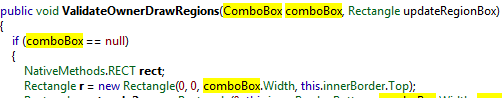
The output of Clousot is a list of warnings and of program repairs.

In theory, Clousot can report all the warnings it finds. This will easily overwhelm the programmer. In practice, Clousot has different, orthogonal features to reduce the noise from warnings. First, it has a baseline feature: only warnings not in the baseline are showed to the user. Second, it provides a selective way to shut off the warnings (at assembly/type/method level, only particular classes of warnings …). Third, it ranks warnings. The verbosity of the warnings can be settled by a simple slider:

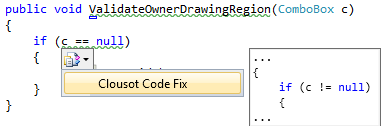


At the warning level “low” only warnings which are definitely a bug (*False*) or that are extremely likely to be a bug are reported. The noise ratio of “low” is extremely low, if not zero. For instance, in Microsoft Dynamics, Clousot with the “low” option reported exactly 29 warnings, and all of them were previously unknown bugs.

We also tried it on some of the .NET system libraries and we found 14 new bugs. For instance, this one is from System.Windows.Forms.dll (v4.0):



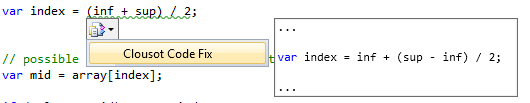
Clousot not only captures the bug, but also proposes a fix for it:



In general, Clousot suggests automatic and verified code repairs for the defects it detects. In our default usage scenario, ***the warning report is set to low and code repairs are on***. Code repairs are either missing contracts (preconditions, object invariants or missing postconditions or implicit code assumptions) or *semantic* fixes to the program source.

###### Example. In the binary search, Clousot detects that when the input parameter array is null, a null pointer exception will definitely be thrown. As a consequence, it suggests adding the precondition:

###### For the overflowing expression, Clousot suggests rewriting it into a (mathematically) equivalent, yet not overflowing form:



###### Please note that Clousot automatically found similar bugs in System.dll and mscorlib (v4.0), which would have been avoided with the repair code functionality.

###### For the possible off by one, Clousot points out that the problems comes from the initialization of sup. If the user changes the initialization of sup to array.Length-1, then Clousot infers the loop invariant

0 ≤ inf ≤ sup < array.Length

###### This it is used to prove that index < array.Length, so that no off-by-one will ever occur.

The repairs suggested by Clousot are inferred from the facts it discovers. They are correct by construction and are both modular and static (no need to run the program or to have test cases). The code repairs that Clousot suggests are: missing preconditions, missing postconditions, missing object invariants, implicit assumptions, wrong initialization, wrong test guards, off-by-ones, incorrect floating point comparisons, and for overflowing integral expressions. Clousot leverages the Roslyn infrastructure to apply code repairs to the source.

### Inter-method inference

When Clousot is done with the analysis of a method, it propagates the inferred preconditions and postconditions to the callers that method. Propagation is useful to reduce the annotation overhead and to help the user start using Clousot on an un-annotated code base, significantly reducing the number of spurious warnings.

# Status

Clousot is part of the CodeContracts tools. CodeContracts tools have been available for download for almost three years now on the DevLabs website. So far, we count almost 60,000 downloads. There is an active external community using Clousot. The community has provided very useful suggestions, feedback and bug reports. Thanks to the community input Clousot has become more stable, precise, and usable. We know of several open source projects using it, *e.g.,* the Facebook SDK for C#.

Clousot is very robust: we recently applied Clousot to Microsoft Dynamics X++, a .dll of 80Mb, containing almost 700,000 methods and 15,596 classes. Clousot analyzed it for 23 hours and found new bugs. This is an email of Peter Villadsen, the PM in X++, sent to his management:

**From:** Peter Villadsen   
**Sent:** Monday, February 06, 2012 3:02 PM  
**To:** Vikram Nagaraj (DYNAMICS); Azfar Moazzam; Richard Barnwell; Tom Ball  
**Cc:** Jay Pillai; Francesco Logozzo; Jakob Steen Hansen; Tanmoy Dutta  
**Subject:** Results from running code contracts on our application assembly

This mail is a followup on the code contract investigation that I have been pursuing on the side for a few weeks.

As you may remember, I started experimenting with Code contracts over the Christmas holiday (a mail was sent to a broad audience at that time – This mail is enclosed for your convenience): The dream scenario is that we would have an automatic way of validating our X++ application code.  The quality of our application code assets is of paramount importance to our success (not least in the cloud), and any means we can use to improve that quality seems warranted. The code contract framework is a framework that can be used to do in-depth analysis of source code, based on proving source code properties. This tool is shipped as part of .NET 4. As it happens, adding contracts in X++ can easily be done (as demonstrated by a prototype shown to Jay and Jakob), because we already have an IL story, and the tool that does code contract validation reads IL assemblies to do its work.

I started validating the approach by annotating some C# code (the XLNT framework) that I know well. I was able to identify 4 bugs with a few hours’ work, which I thought was encouraging. This result leads me to believe that we should not undertake writing C# code without using code contracts: The small extra time spent putting in the assumptions and assertions into the code is largely offset by the benefits reaped from better quality code. However, the real test was to see if the code contract analysis tool is able to realistically deal with the huge DLL that contains all the business logic (the assembly contains close to 700.000 methods). I gave the artifacts to Francesco, who ran it through his tool, which gave us a few interesting results (see below). Note that this assembly is the raw assembly we use for production; no code contracts were explicitly added to it. There is no doubt in my mind that adding Assertions and Assumption in the X++ code will improve the quality tremendously.

The tool ran for around one day, and spit out hundreds of thousands of errors, even without any specific contracts entered by the developers. You can find the log of more than 332000 possible issues here: <http://dynamics/AX/AX7/Teams/BIDevTools/Shared%20Documents/Programming/warnings.txt>. Please see below for some examples. Note, that once the tool has run for the first time, rerunning it (even with a modified assembly) takes around 1.5 hours: In other words a cache is maintained. It is obvious, that the tool would be much more useful if we use feature where the developer provides the list of artifacts that he is interested in, and the tool will run in more manageable time.

Please note, that the results described here have materialized through Francesco’s diligence and his drive to improve their tools in the face of pathological input.

Example 1: The loop below will never run more than once, because the ExtractForm static method invariably throws an exception

**Method 170 : Dynamics.Ax.Application.SysXLNTMetadataExtractor.ExtractForms(System.String)**

Dynamics.Ax.Application.SysXLNTMetadataExtractor.ExtractForms(System.String)[0x15]: warning : Possibly calling a method on a null reference 'local\_1'

Dynamics.Ax.Application.SysXLNTMetadataExtractor.ExtractForms(System.String)[0x58]: reference use unreached

The highlighted code is dead code as ExtractForm always thrown an exception

public static [void](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Void) [**ExtractForms**](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.SysXLNTMetadataExtractor/ExtractForms(String))([[In](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Runtime.InteropServices.InAttribute/.ctor())] [string](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.String) directory)

{

[string](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.String) **nullString** = [PredefinedFunctions](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions).[GetNullString](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions/GetNullString():String)();

    nullString = @"\Forms\";

[TreeNode](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode) **formNode** = [TreeNode](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode).[findNode](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode/findNode(String):Dynamics.Ax.Application.TreeNode)(nullString).[Aotfirstchild](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode/Aotfirstchild():Dynamics.Ax.Application.TreeNode" \o "TreeNode Dynamics.Ax.Application.TreeNode.Aotfirstchild();  CTRL+Click to open in new tab.)();

    if ([TrueFalseHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper" \o "Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper  CTRL+Click to open in new tab.).[TrueFalse](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper/TrueFalse(Int32):Boolean)(([int](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Int32)) ![TrueFalseHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper" \o "Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper  CTRL+Click to open in new tab.).[TrueFalse](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper/TrueFalse(Boolean):Boolean" \o "bool Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper.TrueFalse(bool);  CTRL+Click to open in new tab.)([WinAPI](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.WinAPI" \o "Dynamics.Ax.Application.WinAPI  CTRL+Click to open in new tab.).[folderExists](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.WinAPI/folderExists(String):Boolean)(directory))))

    {

[WinAPI](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.WinAPI).[createDirectoryPath](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.WinAPI/createDirectoryPath(String):Boolean" \o "bool Dynamics.Ax.Application.WinAPI.createDirectoryPath(string);  CTRL+Click to open in new tab.)(directory);

    }

    while ([TrueFalseHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper" \o "Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper  CTRL+Click to open in new tab.).[TrueFalse](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper/TrueFalse(Boolean):Boolean)(![EqualHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.EqualHelper" \o "Microsoft.Dynamics.Ax.Xpp.EqualHelper  CTRL+Click to open in new tab.).[Equal](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.EqualHelper/Equal(Microsoft.Dynamics.Ax.Xpp.XppObjectBase,Object):Boolean" \o "bool Microsoft.Dynamics.Ax.Xpp.EqualHelper.Equal(XppObjectBase, object);  CTRL+Click to open in new tab.)(([XppObjectBase](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.XppObjectBase" \o "Microsoft.Dynamics.Ax.Xpp.XppObjectBase  CTRL+Click to open in new tab.)) formNode, null)))

    {

[ExtractForm](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.SysXLNTMetadataExtractor/ExtractForm(Dynamics.Ax.Application.TreeNode,String)" \o "void Dynamics.Ax.Application.SysXLNTMetadataExtractor.ExtractForm(TreeNode formNode, string directory);  CTRL+Click to open in new tab.)(formNode, directory);

        formNode = formNode.[Aotnextsibling](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode/Aotnextsibling():Dynamics.Ax.Application.TreeNode)();

    }

}

|  |
| --- |
| public static [void](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Void) [**ExtractForm**](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.SysXLNTMetadataExtractor/ExtractForm(Dynamics.Ax.Application.TreeNode,String))([[In](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Runtime.InteropServices.InAttribute/.ctor())] [TreeNode](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.TreeNode) formNode, [[In](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.Runtime.InteropServices.InAttribute/.ctor())] [string](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.0.0:b77a5c561934e089/System.String) directory)  {  [PredefinedFunctions](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions).[LogCQLFuncError](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions/LogCQLFuncError(Int32,String,String))(0x93, "SysXLNTMetadataExtractor", "ExtractForm");      throw new [InvalidRemoteCallException](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.InvalidRemoteCallException/.ctor())();  } |

Example 2: Call on a reference that is provable to be null:

This seems a definite null pointer exception:

   if ([TrueFalseHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper" \o "Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper  CTRL+Click to open in new tab.).[TrueFalse](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper/TrueFalse(Boolean):Boolean)(\_createForDefaultAccount))

    {

[ListEnumerator](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.ListEnumerator) **listEnumerator**;

        if ([TrueFalseHelper](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper" \o "Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper  CTRL+Click to open in new tab.).[TrueFalse](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.TrueFalseHelper/TrueFalse(Boolean):Boolean)(listEnumerator.[Movenext](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.ListEnumerator/Movenext():Boolean)()))

        {

            throw [XppExceptions](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.XppExceptions).[Throw](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.XppExceptions/Throw(Int32):Microsoft.Dynamics.Ax.Xpp.XppException)(([int](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://mscorlib:4.0.0.0:b77a5c561934e089/System.Int32)) [AifFault](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.AifFault).[fault](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Dynamics.Ax.Application:6.0.947.9040/Dynamics.Ax.Application.AifFault/fault(String,String):Dynamics.Ax.Application.Exception)([PredefinedFunctions](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions" \o "Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions  CTRL+Click to open in new tab.).[LookupLabel](http://127.0.0.1/roeder/dotnet/Default.aspx?Target=code://Microsoft.Dynamics.AX.Xpp.Support:6.0.0.0:31bf3856ad364e35/Microsoft.Dynamics.Ax.Xpp.PredefinedFunctions/LookupLabel(String):String)("@SYS326452"), "DimensionContainerNotValidForDefaultAccount"));

        }

    }

More examples in

**Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)**

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0x53]: warning : Possibly calling a method on a null reference '\_ledgerAccount'

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0xd0]: warning : Possibly calling a method on a null reference 'local\_4'

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0x17b]: warning : Calling a method on a null reference 'local\_9'

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0x1e3]: warning : Possibly calling a method on a null reference 'local\_5' (Fixing this warning may solve one additional issue in the code)

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0x1ef]: warning : Possibly calling a method on a null reference 'local\_5'

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0xef]: warning : Possibly calling a method on a null reference

Dynamics.Ax.Application.DimensionServiceProvider.buildDimensionStorageForLedgerAccount(Dynamics.Ax.Application.LedgerAccountContract, System.Boolean, System.Boolean)[0xf9]: warning : Possibly calling a method on a null reference

Best Regards

Peter Villadsen  
Senior Program Manager - X++ experience team   
Microsoft Corporation

phone: +1 425-704-9538   
email: [peter.villadsen@microsoft.com](mailto:peter.villadsen@microsoft.com)  
blog: <http://blogs.msdn.com/x/rss.xml>