z/TPF V1.1

TPF Users Group Fall 2008
Title: z/TPF Visibility
Subtitle: Optimizing your z/TPF programs

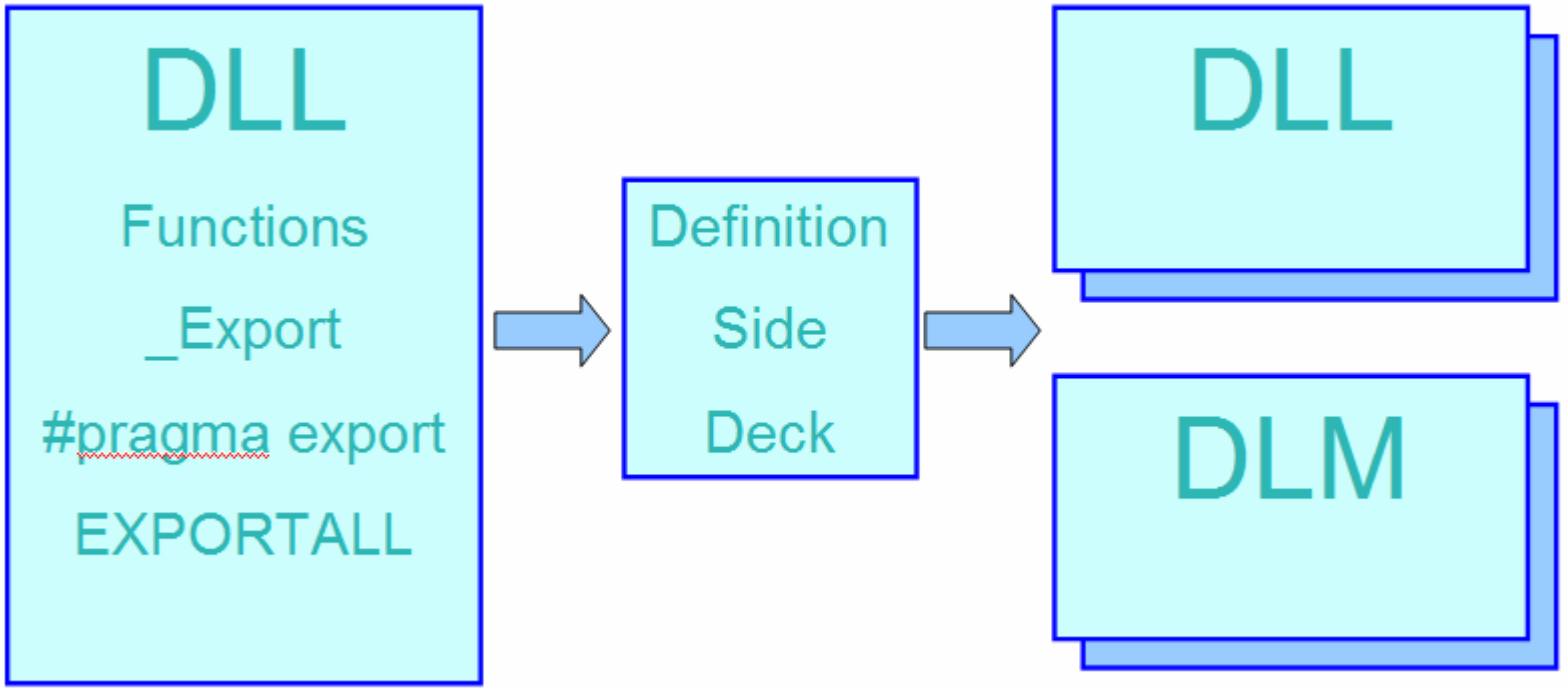
Name: Edwin van de Grift
Venue: Open Source Subcommittee

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Contents

- **Where we came from**
- **Moving forward**
- **Visibility**
  - Why visibility matters
  - Which tools you have to manage visibility
  - How to use the visibility tools

Trip Down Memory Lane



From TPF41 to z/TPF

- **In TPF41**
  - Functions need to be exported to be callable
- **In z/TPF**
  - Functions need to be visible to be callable

What's Our Goal ?

- **Everything that needs to be callable must be visible**
- **Everything that does not need to be callable should be hidden**
- **But there are a few things – especially in C++ – you must be aware of**

Why Does This All Matter?

- **Performance**
  - Size
    - Smaller symbol tables
    - Less generated code
  - Execution time
    - Shorter shared object startup path
    - Faster linkages

Visibility Approaches

- **Two Approaches**
  - Declare everything visible
    - And hide what shouldn't be visible
    - Preferred, and typically easier
  - Declare everything hidden
    - And make callable symbols visible
    - Have to be careful in C++

Visibility in C

- **Use static**
  - This defines a function or variable to be in file or local scope.

Visibility in C++

- **Step 1: Define global visibility**
  - -fvisibility=default
    - Everything is visible
  - -fvisibility=hidden
    - Everything is hidden
  - -fvisibility-inlines-hidden hides inline functions

Visibility in C++

- **Step 2: Define visibility per symbol**
  - __attribute__((visibility("*option*")))
    - "default"
      - Symbol is visible
    - "hidden"
      - Symbol is hidden

Visibility Considerations in C++

- **C++ classes have a number of vague linkage entities**
  - For example typeinfo and vtable
- **These take on the visibility attribute of the class.**
  - If a class is hidden, these entities are hidden
  - This means for example that you cannot
    - Inherit from that class in a different Shared Object
    - Be thrown by an exception

Say What???

- **If you compile the following with –fvisibility=hidden**

```
class Thing {
public:
   __attribute__((visibility("default")))
   Thing();
   __attribute__((visibility("default")))
   ~Thing();
   __attribute__((visibility("default")))
   void doThis();
   void doThat();
private:
};
```

- **There are things you will NOT be able to do, like**
  - Inherit from the class
  - Throw the class

Visibility & MakeTPF

- **APP_EXPORT**
  - ALL (Default) -fvisibility="default"
  - ENTRY -fvisibility="default"
  - LIST -fvisibility="default"
  - VISIBILITY -fvisibility="hidden"
-fivisibility-inlines-hidden for C++

Using APP_EXPORT

- **ALL**
  - Shared Objects with more than 1 callable function
    - TPF41 DLLs & LLMs
- **ENTRY**
  - Shared Objects with 1 callable function
    - TPF41 DLMs & Assembler Programs
- **LIST**
  - Shared Objects that have Assembler functions that need to be hidden
    - TPF41 LLMs
- **VISIBILITY**
  - Shared Objects with more than 1 callable function, if you want to use the opposite approach
    - TPF41 DLLs & LLMs

APP_EXPORT:=LIST

- An export file will be created -- with all symbols defined as visible -- in the build directory if one doesn't exist already
  - ABCD.exp
- When new functions are added to a shared object, makeTPF will create a new version of the export file in the build directory that has the new symbol(s) defined as visible
- The export file needs to be edited and moved from the build directory to the exp directory

```
{
global:
  create_bill;
local:
  calculate_tax;
  print_receipt;
};
```

Example 1 – C Code

abcd.mak

```
APP := ABCD
APP_EXPORT := ALL
C_SRC := program.c
```

program.h

```
void callableFunction();

void anotherCallableFunction();

static void uncallableFunction();

__attribute__ ((visibility("hidden")))
void anotherUncallableFunction();
```

Example 2 – Shared Object with One Entry Point Function

abcd.mak

```
APP := ABCD
APP_EXPORT := ENTRY
APP_ENTRY := ABCD
C_SRC := abcd.c
```

abcd.c

```
int ABCD() {
   return 0;
}

void functionOne(void) {}

static void functionTwo(void) {}
```

Example 3 – C++ Code

abcd.mak

```
APP := ABCD
APP_EXPORT := ALL
CXX_SRC := abcd.cpp
```

abcd.cpp

```
class Thing {
public:
   Thing();
   ~Thing();
   void doThis();
   __attribute__((visibility("hidden")))
   void doThat();
private:
};
```

Example 4 – C in TPF41 Versus z/TPF

```
#pragma export(bopen)
int bopen(const char* file) {
   ...
}

int _Export bclose(int) {
   ...
}
```

```
__attribute__((visibility("default")))
int bopen(const char* file) {
   ...
}
__attribute__((visibility("default")))
int bclose(int) {
   ...
}
```

```
#pragma GCC visibility push(default)
int bopen(const char* file) {
   ...
}
int bclose(int) {
   ...
}
#pragma GCC visibility pop
```

Example 5 – C++ in TPF41 Versus z/TPF

```
class _Export Thing {
public:
   Thing();
   ~Thing();
   void doThis();
   void doThat();
private:
};
```

```
__attribute__((visibility("default")))
class Thing {
public:
   Thing();
   ~Thing();
   void doThis();
   void doThat();
private:
};
```

Example 6 – Single Source

```
#ifndef __370__
   #define _Export
   #define _Export_zTPF __attribute__((visibility("default")))
#else
   #define _Export_zTPF
#endif
```

Toolkit 3.2.7!

OTRKYWDc

```
_Export_zTPF class _Export Thing {
public:
   Thing();
   ~Thing();
   void doThis();
   void doThat();
private:
};
```

Q&A

- **Questions?**

Trademarks

- **IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.**
- **Notes**
- **This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.**
- **Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.**
- **This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.**