PDT's PDB file format definition

Name

PDB definitions --

Definitions

Each item is decribed by a block of lines
First line is always [id] [name] where [id] consists of prefix as described aboved followed by '#' followed by an unsigned (>=0) and unique number (unique inside the item category). The numbering has no other constraints (especially contiguous allocation or start with 0)
Each other line is of the form: attribute value value...
Attributes and values are separated by exactly one space (" ")
Item description blocks are terminated by an empty line
Values which are references to other items are done using IDs
Values are never enclosed in ""
Attributes with boolean value are only listed if true (value == T) i.e. absence means false
String "[...]" after attribute means "0 or more" instances of this attribute line possible

Name

File Header -- Header information included in each pdb file.

<PDB [version]>
lang [language]

Definition

`[version]` which version of PDB file format this file implements, version 3.0 includes statement level information.
`[language]` the programing language that was parsed to generate this file.

Name

Source Files -- This tag records the each pdb files.

so#[id]
[ssys]
sinc so#[id]
[...]
scom co#[comID] [lang] {Location} {Location} [comment]
[...]

Definition

`[id]` Unique number identifying this source file (used to refer to this file elsewhere.)
`[ssys]` ssys tag is printed if this file is a system include, otherwise ommit this line.
`[id]`Id. of the files included by this file.
`[comID]` Unique number identifying this comment in the source file. `[lang]` the programing language that was parsed to generate this file.
`{Location} {Location}` Two location tags, the start of the comment and the end of the comment.
`[comment]` the contents of this comment.

Name

Routines -- This tag records each routine declared in the source file.

ro#[id] [name]
rloc {Location}
rsig ty#[type]
rlink [link]
rkind [kind]
rvirt [virtual]
rgroup gr#[groupId]
racs [access]
rnspace na#[namespaceId]
rroutine [parentRoutineId]
ralias ro#[aliasId]
rimpl ro#[implementsId]
[rstatic]
[rcrvo]
[rinline]
[rexpl]
rtempl te#[templateId]
[rspecl]
[rarginfo]
[rrec]
[riselem]
rstart {Location}
rcall ro#[calleeId] [virtualCall] {Location}
[...]
rret {Location}
[...]
rstop {Location}
[...]
rbody st#[body]
{Statements}
[...]
rpos {Location} {Location} {Location}
{Location}

Definition

`[id]` Unique number identifying this routine. `[name]` The name given this routine in the source file.
`{Location}` The location of this routine in the source file.
`[type]` The id of the tag in the pdb file that defines the type of this Routine.
`[link]` The kind of link this is either: NA, INTERNAL, C, C++, FINT, F90.
`[kind]` The kind of Routine this is either: NA (not defined), ext (external), tproto (template/protoype), fext (Fortran external), fprog (Fortran program), fbdat (Fortran block data), fint (Fortran internal), fstfu (Fortran statement function), fintrin (Fortran intrinsic), fmproc (Fortran module procedure), funspec (Fortran unspecified).
`[virtual]` Either yes or no.
`[groupId]` The Id. for this routines parent class.
`[access]` The access level for this routine.
`[namespaceId]` Id. for the Parent namespace.
`[parentRoutineId]` Id. for the Parent routine.

[aliasId] Id. for this routines alias (through interface).
[implementsId] Id. for the routine this routine implements (for Fortran alaised functions.)
[rstatic] rstatic if this routine is static. (Ommit line otherwise)
[rcrvo] rcrvo if this routine has a covarient return virtual override
[rinline] rinline if this routine is inlined, ommit this line otherwise.
[rexpl] rexpl if this routine is explicit ctor ommit otherwise.
[templateId] Id for the template, if this routine implements one, ommit otherwise.
[rspecl] rspecl if this routine is specialized
[rarginfo] rarginfo if this routine explicatly defined interface.
[rrec] rrec if this routine is declared recursive.
[riselem] riselem if this routine is elemental.
{Location} The location of the first executable statement.
[calleeId]The Id. that defines the type of the routine this routine calles.[virtualCall] Is this call virtual. {Location} The location of this routine call in this routine.
{Location} Location of the return statements.
{Location} The location of the stop statements.
[body]Id. of the first statement in the body of this Routine.
{statements} Each statement inside the body of this Routine.
{Location} {Location} {Location} {Location} Four location tags, in order, the beginning of the Routine declaration, the ending of the Routine declaration, the begining of the Routine body, and lastly the ending of the Routine body.

Name

Types -- This tag records each Type declared in the source file.

ty#[id] [name]
ykind [kind]
yrett ty#[return type]
yargt ty#[argument type]
yexcept ty#[exception type]
yfkind [float type]
yikind [integer type]
yclen [length]
yshape [shape]
yloc {Location}
ygroup gr#[groupId]
yacs [access]
[ysigned]
yenum [name] [value]
[...]
yptr (ty#|gr#)[pointerId]
yref (ty#|gr#)[referenceId]
[yellip]
yqual [qualifier]
yelem (ty#|gr#)[elementId]
[ystat]
ynelem [number of elements]
yrank [rank]
ydim [dimension]
ytref (ty#|gr#)[typedefId]
ympgroup gr#[ptr_groupId]
ymptype (ty#|gr#)[ptr_typeId]

Definition

`[id]` Unique number identifying this Type. `[name]` The name given to this Type in the source file.
`[kind]` The kind of Routine this is, either: NA (not defined), bool (boolean), enum (enumeration) , err (error/exception), func (function), void, int (integer), float, ptr (pointer), ref (reference to memory location), array, tref (template reference), ptrmem, traram, wchar, c_type, ferr (Fortran error), fvoid (Fortran void), fint (Fortran integer), flogic, ffloat (Fortran float), ffunc (Fortran function), fchar (Fortran character), farray (Fortran array), fcmplx (Fortran complex), funspecfunc, fbldat, fmod, fptr, f_type, group.
`[return type]` Id. of the tag that defines the return part of this type (Ommit line if type is not a function type).
`[argument type]`Id. of the tag that defines the parameter part of this type (Ommit line if type is not a function type).
`[execption type]`
`[float type]` The type of this float, either, float, dbl (double), or longdbl (long double). (Ommit line if this type is integer or is non-primitive.)
`[integer type]` The type of this integer, either char (charater), schar (short charater), uchar (unsigned charater), short, ushort (unsigned short), int (integer), uint (unsigned integer), long, ulong (unsigned long), longlong, ulonglong (unsigned long long). (Ommit line if this type is float or is non-primitive.)
`[length]` The length of the array (Ommit line if type is not array.)

[shape] The shape of this Fortran array, either, explicit (set when the rank and extent are defined explicitly), asize (set when the extend of one or more dimension is undefined), ashape (set when the rank of an array is left undefined), deferred (set when an array is allocated but undefined). (Ommit line if type is not a fortran array.)
{Location} Location of where this type is defined in the source code.
[groupId] This type's parent group.
[access] This type's access level.
[ysigned] If this type is signed, ommit otherwise
[name] The name of the enumeration member. [value] The value of the enumeration member.
[pointerId] the type or group Id. that this type is pointing to, ommit if this is not a pointer type.
[referenceId] the type or group Id. that this type is referencing, ommit if this is not a reference type.
[yellip] yellip if this type has ellipsis, ommit line otherwise.
[qualifier] the qualifiers this type has, if any.
[elementId] the type or group of the elements in the array, ommit if this is not an array type.
[ystat] ystat if this type is static.
[ynelem] The number of element in this array, -1 if this array has variable length, ommit line if this is not an array type.
[rank] The rank of this array, ommit line if this is not an array type.
[dimension] The dimension of this array, ommit line if this is not an array type.
[typedefId] The typedef type if there is one.
[ptr_groupId] The type of group to which the type or group that is pointed to belongs.
[ptr_typeId] The type that the type or group that is pointed to belongs.

Name

Locations -- This simple tag is used to store a location in a source file. Used only within other tags, not found by itself.

so#[id] [line number] [colunm number]

Definition

`[id]` The id number of the source file. `[line number]` The line number of the location. `[colunm number]` The colunm number of the location.

Name

Statements -- This simple tag is used to record a statement in the source file. Used only within other tags, not found by itself.

stmt st#[id] [name] {Location} {Location} [next] [down]

Definition

`[id]` the unique number identifying this statement. `[kind]` the kind of statement, either, na (not defined), switch, case, init, return, if, empty, for, goto, continue, break, label, block, asm, expr, assign, throw, while, do, try, catch, decl, set_vla_size, vla_decl, vla_dealloc, fallocate, fassign, fio, fdo, fdeallocate, freturn, fif, fgoto, fsingle_if, fstop, or flabel. `{Location} {Location}` Two locations, where the statement begins and where it ends. `[next]` Id number of the next statement (NA if there is none). `[down]` Id number of the statement beneath this one, (ie, if this were a block statement, then this would be the first statement within the block.) (NA if there is none).

Name

Groups -- This simple tag is used to record a group (classes in C) in the source code.

gr#[Id] [name]
gloc {Location}
ggroup gr#[groupID]
gacs [access]
gnspace [namespaceID]
gkind [kind]
gtempl te#[templateID]
[gspecl]
[gsparam] [type] (ty#|gr#)[paramId]
[...]
gbase [virtual] [access] gr#[baseId] {Location}
[...]
gfrgroup gr#[friendId] {Location}
[...]
gfrfunc ro#[routineID] {Location}
[...]
gfunc ro#[member_routineId] {Location}
[...]
gmem [members]
[...]
gmloc {Location}
gmgroup gr#[parentId]
gmacs [member_access]
gmkind [member_kind]
gmtype (ty#|gt#)[member_type]
gmtempl te#[templateId]
[gmspecl]
[gmconst]
[gmisbit]
[gmmut]
[gmconst]
[gmtempl] te#[templateId]
gpos {Location} {Location} {Location} {Location}

Definition

`[id]` The unique number identifying this group.
`[name]` The name of this group.
`{Location}` Location where the group header begins.
`[groupId]` The group Id. of this group's parent.
`[gacs]` This group's access level.
`[namespaceId]` This group's parent namespace.
`[kind]` The kind of groups this is.
`[templateId]` The id of the template this group implements, if it implement one.
`[gspecl]` gspecl if this group is specialized.
`[type]` The type of this parameter. `[paramId]` the id. for this parameter's type.
`[virtual]` vir if the direct base group is virtual. `[access]` the access level for the base group. `[base_groupId]` the id of this groups base group. `{Location}`
`[friendId]` The Id. of the friend groups of this group. `{Location}` Location of this friend group.

[routineId] The Id. of the friend functions of this group. {Location} Location of this friend function.
[member_routineId] the Id. for the routines that are members of this group. {Location} The location of this member routine.
[member] The name of the non-function members of this group.
{location} the location of this member.
[parentId] the parent group of this member.
[member_access] the access level of this member group.
[member_kind] the kind of the member this is.
[member_typeId] the type id of this member
[member_templateId] this member's templete Id, if it implements one.
[gmspecl]gmspecl if this member is specialized.
[gmconst] gspecl if this member is initalized.
[gmisbit] gmisbit if this member is a bit field.
[gmmut] gmmut if this member is mutable.
{Location} {Location} {Location} {Location} Four location tags, in order, the beginning of the Group declaration, the ending of the Group declaration, the begining of the Group body, and lastly the ending of the Group body.

Name

Templates -- These tags record templates in the source files.

te#[templateId]
tloc {Location}
tgroup gr#[groupId]
tacs [access]
tnspace na#[namespaceId]
tdecl te#[declareId]
tdef te#[definitionID]
tkind [kind]
tparam [type_name] ty#[typeId]
[...]
tsparam [...] [param_name] (ty#|gr#)[paramId]
[...]
tproto ro#[routineId]
ttype (ty#|gr#)[variable_typeId]
ttext [template_text]
tpos {Location} {Location} {Location} {Location}

Defintions

`[templateId]` The unique number identifying this template item.
`{Location}` The location where this template definition starts.
`[groupId]` the Id. of this template's parent group.
`[access]` this template access level.
`[namespaceId]` this template's parents namespace.
`[declareId]` The Id of the template which declares this one, if one does.
`[definitionId]` The Id of the template which defines this one, if one does.
`[kind]` The kind of template this is.
`[type_name]` The type of any template parameters. `[typeId]` The Id of the type that defines this parameter.
`[param_name]` The name of this template parameter. `[parameterId]` The Id. of the type or groups that defines this template parameter.
`[routineId]` The Id. of this template's prototype instantiation.
`[variable_typeId]` The Id. of the type that defines this variable.
`[template_text]` The actual text of this template.
`{Location} {Location} {Location} {Location}` Four location tags, in order, the beginning of the Template declaration, the ending of the Template declaration, the begining of the Template body, and lastly the ending of the Template body.

Name

Namespaces -- These tags record the namespaces inside the source file.

na#[namespaceID] [name]
nloc {Location}
nnspace na#[parentId>
nmem (ty#|ro#|gr#)[memberId]
[...]
nalias [alias_name]
npos {Location} {Location} {Location} {Location}

Definition

`[namespaceId]` The unique number identifying this namespace. `[name]` The name of this namespace.
`{Location}` The location where the definition of this namespace starts in the source file.
`[parentId]` The Id. of this namespaces parent namespace.
`[memberId]` The Id. of the type, routine, or group that defines this member of the namespace.
`[alias_name]` The name of this namespaces alias.
`{Location} {Location} {Location} {Location}` Four location tags, in order, the beginning of the Namespace declaration, the ending of the Namespace declaration, the begining of the Namespace body, and lastly the ending of the Namespace body.

Name

Macros -- These tags record each occurance of a marco in the source file.

ma#[macroId] [name]
mloc {Location}
mkind [kind]
mtext [text]

Definition

`[macroId]`
`{Location}` The location in the source file where this macro begins.
`[kind]` The kind of macro this is, either defined or undefined.
`[text]` The actual text of this macro.

Name

Pragmas -- These tags record every Pragma within the source file.

pr#[pragmaId] [name]
ploc {Location}
pkind [kind]
ppos {Location} {Location}
ptext [text]

Definition

`{Location}` The location in the source file where this pragma begins.
`[kind]` The kind of Pragma this is.
`{Location}{Location}` The location where this pragma begins and ends.
`[text]` The actual text of the pragma.